Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00484/2009
Datum: 15.10.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NEJC ŠUŠTARŠIČ**

Naslov: **KONFIGURIRANJE RAČUNALNIŠKO KRMILJENEGA TOČILNEGA SISTEMA**

CONFIGURING THE COMPUTER-CONTROLLED BAR SYSTEM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

V nalogi predstavite računalniško krmiljene točilne sisteme in njihovo povezavo s POS sistemi.
Opišite protkol za konfiguracijo in izdelajte aplikacijo za konfiguriranje točilnega računalnika.

Mentor:
pred. mag. Igor Škraba

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:
prof. dr. Franc Solina

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Nejc Šuštaršič

KONFIGURIRANJE RAČUNALNIŠKO KRMILJENEGA TOČILNEGA SISTEMA

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Ljubljana, 2010

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Nejc Šuštaršič

KONFIGURIRANJE RAČUNALNIŠKO KRMILJENEGA TOČILNEGA SISTEMA

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: mag. Igor Škraba

Ljubljana, 2010

ZAHVALA

Iskreno se zahvaljujem mag. Igor Škraba za usmerjanje pri izdelavi diplomske naloge. Posebno zahvalo namenjam tudi svojemu očetu, ki mi je v podjetju omogočil izdelavo tovrstne naloge ter mi predstavljal knjižnico znanj z danega področja ter mami, ki me je vsa leta študija spodbujala.

KAZALO

SEZNAM UPORABLJENIH KRATIC IN SIMBOLOV

POS - Point of sale - blagajniško mesto

XML - Extensible markup language - razširljiv označevalni jezik

UPS - Uniterruptible power supply - brezprekinitveni napajalnik

BARCODE - optična prezentacija podatkov o identifikaciji

LAN - Local area connection - lokalna mreža

RFID - Radio frequency identification - tehnologija identifikacije na podlagi radijskega signala

COM - vmesnik za serijska vrata

.NET Framework – Microsoftovo programsko ogrodje

VisualBasic – Programski jezik

POVZETEK

Glavni cilj diplomske naloge je izdelava aplikacije za konfiguriranje računalniško krmiljenega točilnega sistema za gostinstvo. V uvodnem delu naloge sem predstavil teoretična izhodišča s področja blagajniških in točilnih sistemov, ki so potrebna za razumevanje ciljne problematike.

Tako kot na marsikatero drugo področje je tudi na področje upravljanja gostinskih lokalov že davno vstopila računalniška tehnologija. Sestavna dela vsakega gostinskega lokala sta blagajniški sistem ter točilni sistem. Manj zahtevni gostinci se zadovoljijo s preprostimi blagajniškimi sistemi, ki implementirajo le osnovne funkcije. Hkrati ne investirajo izdatno v točilni sistem, ki je tako sestavljen le iz posameznih nepovezanih točilnih naprav. Najpogosteje so to le avtomat za kavo in točilne pipe za pivo. Zahtevnejši gostinci omenjenim sistemov namenijo več pozornosti, čemur sledijo tudi višji finančni vložki. Z njimi je v gostinskem lokalu mogoče uvesti računalniški blagajniški sistem, ki v povezavi z računalniško krmiljenim točilnim sistemom predstavlja zaključeno celoto, ki tako gostincem omogoča celovitejši nadzor nad delovanjem lokala (prodaja, poraba, itd.).

Računalniško krmiljen točilni sistem je potrebno pred delovanjem skonfigurirati. V ta namen sem spisal aplikacijo, ki to nalogo v največji meri poenostavi. Aplikacija omogoča komunikacijo s točilnim računalnikom, ki je bistveni del takšnega točilnega sistema, saj usmerja njegovo delovanje. Podatki s točilnega računalnika so predstavljeni v tabelarični obliki. Tovrsten način omogoča enostaven vnos novih vrednosti, ki jih uporabnik nato zapiše na točilni računalnik. Komunikacija s točilnim računalnikom zahteva tudi opis protokola. Aplikacija omogoča tudi shranjevanje konfiguracije v XML datoteko in branje iz nje.

Pri opisu aplikacije sem najprej opisal osnovne gradnike, to so osnovne procedure in funkcije, s pomočjo katerih sem izvedel večino zahtevanih nalog. Vsak opis vsebuje opis delovanja procedure ali funkcije ter njeno predstavitev v obliki diagrama. Nadaljeval sem z opisom grafičnega vmesnika, kjer sem opisal tudi nekaj zanimivejših primerov spoprijemanja s pisanjem dinamičnega uporabniškega vmesnika in zagotavljanja vnosa pravilnih vrednosti v konfiguracijsko tabelo.

Ključne besede:

blagajniški sistem, POS sistem, računalniško krmiljen točilni sistem

SUMMARY

The principal goal of my diploma thesis is creating an application for configurating computer-controlled beverages dispensing systems. In the preamble of my thesis I present the theoretical platform for point of sale systems and beverages dispensing systems, which are required for the understanding of the target problematics.

As with many other fields, computer tehnologies entered the field of managing bars and restaurants quite some time ago. Basic components of every bar or restaurant are point of sale systems and beverages dispensing systems. Less demanding owners of bars or restaurants are satisfied with simplest of the point of sale systems, which implement only the most basic functions. At the same time they do not invest much in dispensing systems, which are most often made up of only unconnected drink dispensers, such as coffee machines or dispensers for beer. More demanding owners pay more attention to the mentioned systems, which often results in larger investments in them. Thus it si possible to introduce a computerized point of sale system, which in conjunction with beverages dispensing system represents a complete whole that allows the owner a more comprehensive way of controlling the operation of the bar or a restaurant.

A computer-controlled beverages dispensing system needs to be configurated before operating. For this purpose I wrote an application that simplifies this process. The application implements communication with dispensing computer which is the essence of computer-controlled beverages dispensing system. Data from the dispensing computer are reprezented in a tabelaric form. Such manner enables a simple entry of new values, which can then be saved on the dispensing computer. Communication between dispensing computer and any other copmuter demands a protocol, whis is also described in this thesis. The application also allows a configuration to be saved in a XML file or to be read from one.

In the description of the application I initially describe the basic procedures and functions that allow me to accomplish most of the tasks. Every description also contains a flow chart diagram of the procedure or the function. In continuation I describe the graphical user interface of the application. At this point I also explain some interesting cases of dealing with dynamic graphical interface and assurance of entering data of correct values in the configuration table.

Key words:

cash register system, POS system, computer-controlled beverage dispensing system

1 UVOD

Prvi računalniški blagajniški sistemi v gostinstvu, ki so podobni današnjim, so se pojavili že sredi 80-ih let. Osnovne funkcije se z leti niso spreminjale. Te so: sprejem naročil gostov, distribucija naročil na tiskalnike v kuhinji ali za točilnim pultom, izdajanje računov za opravljeno storitev ter običajno tudi določeni izpisi povezani s stanjem blagajne, dnevnim prometom itd. Mnogi ponudniki PC komponente integrirajo v na videz privlačnejše terminale, v katerih so združeni z zaslonom. Z leti, ko so se razvili grafično privlačnejši operacijski sistemi in zasloni na dotik, so slednji na trgu POS sistemov povsem prevladali. Delo na njih je znatno hitrejše kot delo s tipkovnico ali miško. V zadnjih letih pa so se kot učinkovit pripomoček izkazale brezžične naprave, ki omogočajo oddaljen vnos naročil v POS sistem.

POS sistem ni edini sistem, ki je v gostinstvu potreben in zaželen. Precej pozornosti marsikateri gostinec posveti točilnemu sistemu. S točilnim sistemom lahko uvedemo boljši nadzor in kontrolo pri točenju pijač, učinkovitejše hlajenje pijač ter prihranek skladiščnega prostora.

Točilnih sistemov je mnogo vrst in se razlikujejo tako v implementacijah pri različnih proizvajalcih kot tudi v nivoju odprtosti/zaprtosti. Na splošno delimo točilne sisteme na odprte ter zaprte. Pri prvih točilni sistem ni nujno računalniško krmiljen, hkrati pa ne komunicira s POS sistemom. Pri slednjih pa je točilni sistem računalniško krmiljen in komunicira s POS sistemom, od katerega prejema naročila za točenje. POS sistem torej v večji meri pripomore k nadzoru točenja in točnosti podatkov.

Vsak točilni sistem je pred uporabo potrebno konfigurirati. Sam sem zavoljo lažjega, hitrejšega in predglednejšega konfiguriranja točilnega sistema izdelal aplikacijo, ki bo predstavljala prijaznejši, bolj logičen in hitrejši vmesnik za opravljanje tovrstnih nalog pri točilnem sistemu SKAT.

Da bi bralec dobil celovitejši vpogled v tematiko diplomske naloge, se ta začne s kratkim opisom POS sistemov, saj so ti pomemben del zaprtih točilnih sistemov in jih v diplomski nalogi večkrat omenjam. Nadalje bom opisal zasnovo in naloge točilnih sistemov, končal pa bom z opisom svojega pristopa k izdelavi prej omenjene aplikacije. Opisal bom zahteve, izbrano orodje in okolje, komunikacijski protokol, implementacijo osnovnih funkcij aplikacije ter njen grafični vmesnik. Prav tako bom opisal nekaj težav, na katere sem naletel pri pisanju aplikacije, najsi bo njihov izvor v samem orodju ali v točilnem računalniku.

2 POS SISTEMI

Izraz *point of sale* (POS) oziroma *mesto prodaje* je razumljiv. Gre za mesto, kjer se vršijo poslovne transakcije med prodajalcem in kupcem.

Point of sale system (v nadaljevanju POS sistem) je sistem, ki običajno ponuja računalniško vodeno prodajo storitev ali materiala. POS sistem lahko sestoji iz enega ali več POS terminalov oziroma blagajniških mest.

POS sistemi si v sklopu diplomske naloge zaslužijo nekaj pozornosti, saj so sestavni del celovitih točilnih sistemov.

2.1 Enostavni POS sistemi

Najenostavnejši sistemi so elektronske registerske blagajne, ki imajo omejene procesorske in pomnilniške zmogljivosti. Njihov nabor funkcij je posledično precej omejen. Poleg izdaje računov tovrstni sistemi omogočajo tudi preproste dnevne izpise, kot sta dnevna prodaja po artiklih ali izpis stanja blagajne. Običajno je uporabniku vidni del sestavljen iz prikazovalnika zneska računa in fiksnega števila tipk, preko katerih uporabnik vnaša pozicije računa. Poleg tega premore še tiskalnik za izpis računa.

2.2 Kompleksnejši POS sistemi

Že dolgo tehnika omogoča izdelavo kompleksnejših POS sistemov, ki so celovit računalniški sistem. Ne glede na proizvajalca gre običajno za skupek računalniških komponent za osebne računalnike, na katerih najpogosteje teče operacijski sistem (v nadaljevanju OS) DOS, Windows ali Linux ter na njem namenska aplikacija. Na terenu dandanes srečamo še precej sistemov z nameščenim OS DOS. Takšne rešitve so cenejše, tako zaradi starejše programske opreme kot tudi zaradi cenejše strojne opreme, ki je potrebna za nemoteno delovanje sistema.

Blagajniško mesto v gostinstvu sestavljajo naslednje komponente:

- osebni računalnik,
- zaslon (ali zaslon na dotik),
- POS tiskalnik (75-mm ali 80-mm trak; zaželjeni hitrejši termalni tiskalniki),
- tipkovnica (in miška).

Pogojno lahko blagajniškemu mestu dodamo še:

- dodatne POS tiskalnike (kuhinjski tsikalnik, tiskalnik na točilnem pultu ...)
- UPS (Uninterruptible power supply) napravo (za pravilno zaustavitev sistema ob izpadu elektrike),
- čitalec natakarskih ključev (magnetni ključi, rfid zapestnice),
- čitalec BAR kode,
- brezžično napravo za oddaljeno delo z blagajniškim mestom,
- elektronsko tehtnico,
- bančni POS terminal za plačilne kartice.

2.3 Funkcije sodobnih POS sistemov

Sodobni POS sistemi za razliko od registerskih blagajn omogočajo precej več funkcionalnosti. Pomembne ali uporabne funkcionalnosti za uporabo v gostinstvu so (lahko):

- prijava v program na podlagi uporabe ključev (magnetni ključi, RFID zapestnice ...) in evidentiranje delovnega časa,
- vnos naročil, vezanih na mizo (v žargonu temu pravimo boniranje),
- izdaja delnih ali nedeljenih računov, dobavnic in originalnih računov,
- prevzem in prenos miz med natakarji,
- distribucija boniranih naročil na različne tiskalnike ali v kuhinji ali na drugem koncu točilnega pulta,
- izdajanje naročil za točenje točilnemu sistemu,
- povezava z brezžičnimi napravami, ki predstavljajo mobilna blagajniška mesta,
- nastavljanje točilnega sistema,
- razni izpisi (izpis stanja blagajne, izpis obračuna natakarja, izpis prometa itd.),
- vnos dnevnih popisov stanja zalog,
- povezava z elektronskimi tehnicami ali za prodajo blaga ali za kontrolo stanja,
- v primeru nastanitvenih objektov v povezavi z recepcijskim sistemom prenos gostinskih storitev na sobo v recepcijski aplikaciji.

Pomembna lastnost, po kateri se POS sistemi med seboj lahko razlikujejo, je, ali blagajniška aplikacija in aplikacija za materialno poslovanje uporabljata isto bazo ali ne. V gostinstvu praviloma velja, da sta bazi obeh aplikacij ločeni in da prometne podatke prenašamo iz blagajniških mest na mesto, kjer jih obdelamo. Za majhne sisteme z enim blagajniškim mestom je združena reštev primerna. Takšna preprostejša rešitev je cenejša, praviloma zaradi manj zahtevne programske opreme kot tudi zaradi manjšega obsega strojne opreme (obe aplikaciji sta običajno na istem računalniku). Slabost pa je, da moramo določena dela opravljati izven delovnega časa lokala, saj z njimi ne smemo motiti tekočega blagajniškega dela.

Takšen pristop pa ni primeren za večje sisteme, kjer se mnogokrat več blagajniških mest povezuje v skupna obračunska mesta. V tem primeru morajo povezana blagajniška mesta delati nad skupno bazo, česar preprostejše rešitve običajno ne omogočajo. Prometne podatke s posameznih obračunskih mest se nato pošilja aplikaciji za materialno vodenje, ki jih vodi v lastni bazi. V njej načeloma urejamo tudi vse matične podatke (natakarji, artikli, materiali, normativi itd.), predvsem pa na podlagi prometnih podatkov izdelujemo dnevne obračune, letna poročila, vodimo evidenco stanja zalog itd.

Zgornje je najlažje doseči, če vsa obračunska mesta in aplikacijo za vodenje materialnih evidenc povežemo v lokalno mrežo (LAN – Local Area Connection). Dandanes, ko poceni internetni priključki omogočajo velike pasovne širine, je smiselno zagotoviti tudi oddaljen dostop do te mreže, saj je tako mogoče opraviti velik del servisnih ali svetovalnih del.

Povezave s točilnim sistemom ne omogočajo vsi POS sistemi. Hkrati POS sistemi, ki imajo tovrstno komunikacijo realizirano, ne delujejo vsi enako dobro v povezavi z računalniško krmiljenim točilnim sistemom. Gostinec mora zato v fazi odločanja o izbiri ponudnika POS sistema tega temeljito preveriti, če bo POS sistem znal odigrati svojo vlogo. V nasprotnem primeru se lahko draga investicija v zaprt točilni sistem izkaže za zgrešeno.

2.4 Izdajanje naročil za točenje točilnemu sistemu

V sklopu te diplomske naloge je najpomembnejše *Izdajanje naročil za točenje točilnemu sistemu*. Kako poteka? Natakar se na blagajniškem mestu prijavi v sistem in vnese naročila strank na mizo. Naročila je nato možno stornirati ali na njihovi podlagi izdati račun, dobavnico ali originalni račun. Naročila POS sistem hkrati posreduje točilnemu sistemu. Tehnično gledano to storimo na enak način, kot je opisan v 4.1 Protokol DS1. Razlika je v protokolu, in sicer v vrsti in pomenu sporočil, ki si jih sistema izmenjujeta. Ko v točilnem sistemu boniramo pijačo, na ustrezni točilni napravi običajno zasveti k pijači pripadajoča tipka, ki uporabnika opozori, da lahko sproži točenje.

Na sliki 2.1 je prikazano blagajniško mesto, ki vsebuje tudi natakarsko ključavnico.

Slika 2.1: Blagajniško mesto z zaslonom na dotik

3 RAČUNALNIŠKO KRMILJENI TOČILNI SISTEMI

Točilni sistem je v osnovi sistem, namenjen točenju pijač. Poleg točenja ponuja tudi druge pomembne funkcije, ki so opisane v poglavju 3.3 Funkcije točilnih sistemov.

3.1 Dilema odprtih in zaprtih točilnih sistemov

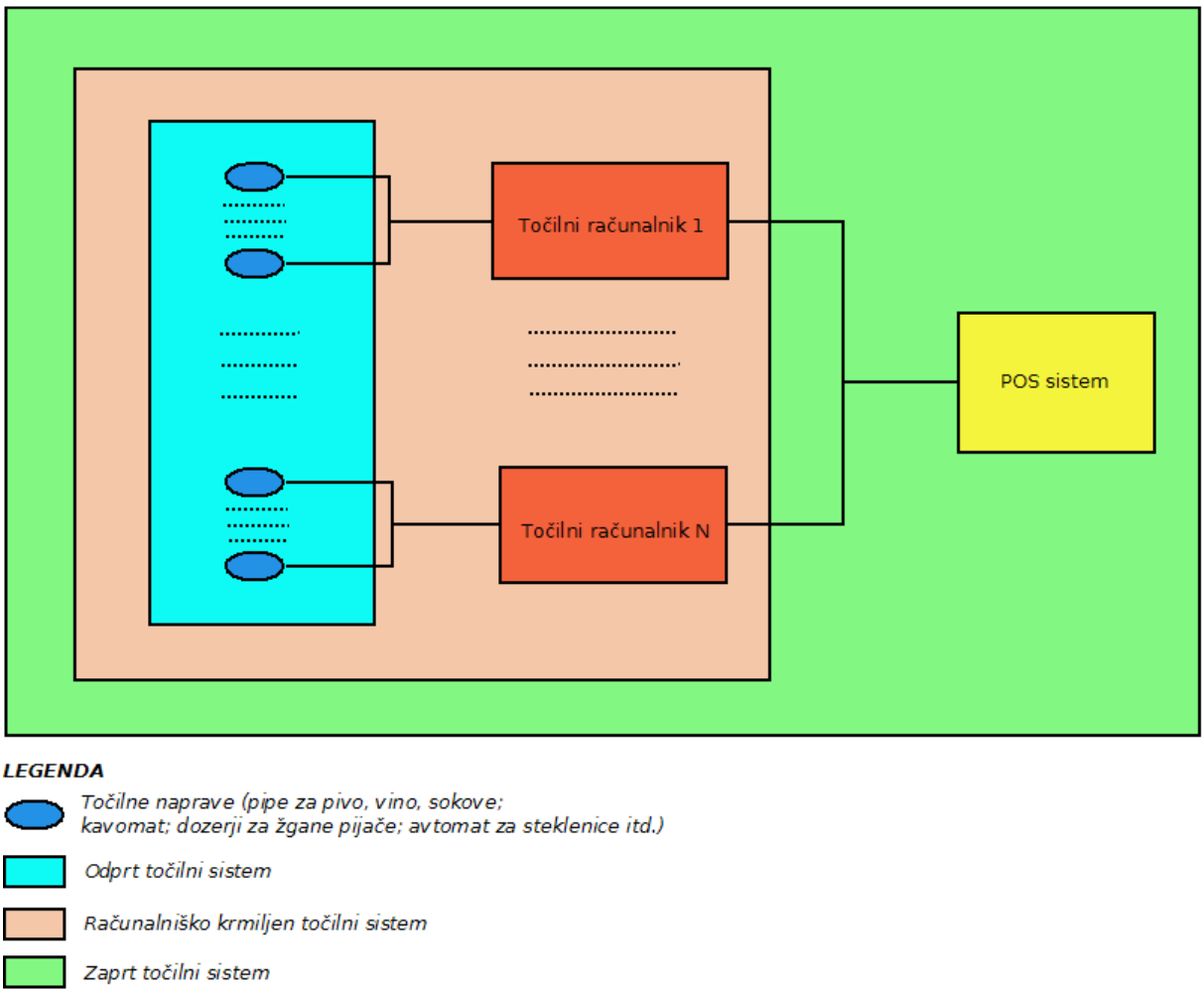
Točilne sisteme lahko delimo glede na nivo nadzora in kontrole, ki ju sistem omogoča, na odprte in zaprte točilne sisteme.

Za odprt sistem velja, da ne omogoča nikakršnega nadzora in kontrole. Zaradi tega je sistem dostopen vsakomur, oziroma je odprt za vsakogar. Edini način nadzora in kontrole je fizična prisotnost nadzorne osebe. Dobra stran tovrstnih sistemov je, da še vedno zagotavljajo hlajenje in skladiščenje. Vendar slaba stran prevlada, saj imamo zaradi pomanjkanja nadzora in kontrole najverjetneje dodatne stroške zaradi porabe, ki presega prodano količino.

Najenostavnejši način uvedbe nadzora točenja in kontrole iztočenih količin je z uvedbo bonirnih in prodajnih števcev. S prvimi nadziramo, koliko katere pijače se sme točiti, oziroma vanje zapisujemo naročene količine pijač. Z drugimi beležimo iztočene količine. Za uvedbo števcev moramo v točilni sistem vpeljati točilni računalnik, s katerim lahko krmilimo posamezne naprave točilnega sistema. Ime prodajni je zavajujoče, saj sistem ne razpolaga s podatkom, ali je iztočena pijača dejansko bila tudi prodana. Pozitivna stran tega je, da sedaj razpolagamo z natančnejšo informacijo o iztočenih količinah. Ostaja pa pomakljivost, da morebitnih razlik med prodanimi in iztočenenimi količinami še vedno ne znamo povezati s konkretno osebo. Še večji se izkaže problem primerjave podatkov, saj je za podatke v prodajnem števcu in prometne podatke iz POS sistema težko zagotoviti enako časovno obdobje, brez katerega primerjava za nas ne pomeni pomembne informacije. Hkrati pa je ta primerjava časovno zahtevna.

Če želimo odpraviti pomakljivosti v zvezi s primerjavo podatkov, je potrebno uvesti še sistem nadzora po principu »kdo naroča«. Za uvedbo tovrstne nadzorne funkcije moramo točilni računalnik povezati s POS sistemom, ki točilnemu sistemu na podlagi boniranih količin delegira, koliko katere pijače sistem sme stočiti. Takšen sistem imenujemo zaprt sistem. Zanj je, analogno odprtemu sistemu, značilno, da se lahko toči le na podlagi naročil, ki jih lahko vnašajo le pooblaščene osebe, oziroma osebe, ki imajo dostop do POS aplikacije.

Na sliki 3.1 je shematičen prikaz razlik med odprtim in zaprtim točilnim sistemom.



Slika 3.1: Shematičen prikaz razlik med odprtim, računalniško krmiljenim in zaprtim točilnim sistemom

3.2 Točilni računalnik

Kot je bilo omenjeno v prejšnjem poglavju, nam funkcije nadzora in kontrole omogoča točilni računalnik. To je računalnik, ki omogoča krmiljenje naprav točilnega sistema.

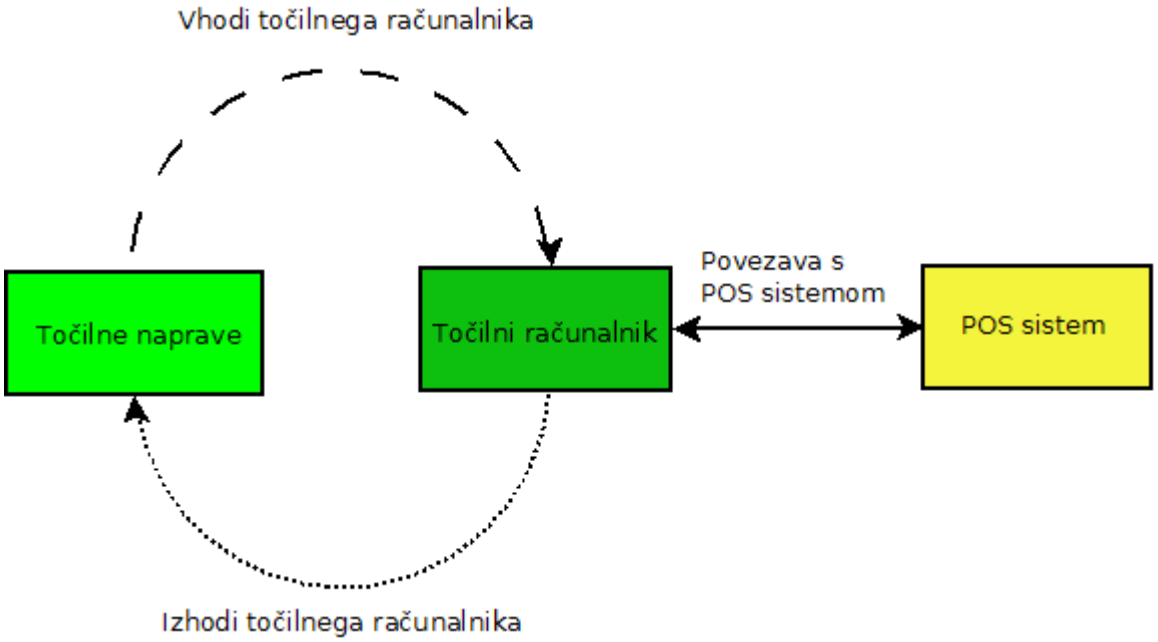
3.2.1 Opis točilnega računalnika

3.2.1.1 Vhodno/izhodni sistem

Bistveni del točilnega računalnika je vhodno/izhodni sistem, preko katerega računalnik komunicira s posameznimi napravami točilnega sistema. Vhode predstavljajo tipke ter turbine, izhode pa releji oziroma polprevodniška stikala za krmiljenje ventilov ali enostavnejše ventili. Število realiziranih vhodov in izhodov je običajno med 100 in 250. Število je odvisno od implementacije točilnega računalnika pri različnih proizvajalcih.

Načeloma točilni sistem potrebuje večje število vhodov kot izhodov, saj imamo za isto vrsto pijače lahko nastavljenih več točilnih količin, ki so povezane na posamezne tipke. Detajlnejši opis razloga za slednje sledi v 3.2.2 Delovanje točilnega računalnika.

Na sliki 3.2 so prikazane povezave točilnega računalnika s točilnimi napravami in POS sistemom.



Slika 3.2: Povezave točilnega računalnika

Točilni računalnik je v relaciji do POS sistema pasiven, saj le izpolnjuje od POS sistema prejete ukaze in nanje odgovarja z rezultati izvedenih ukazov. Za opis odnosa med točilnim računalnikom in POS sistemom lahko uporabimo tudi analogijo MASTER – SLAVE (gospodar – suženj). POS sistem je v tem primeru *master*, točilni sistem pa *slave*.

3.2.1.2 Podatkovna struktura

Za podatkovno strukturo, ki nam omogoča nadzor nad delovanjem točilnega sistema, lahko vzamemo dvodimenzionalno tabelo (»tabela pijač«), v kateri vrste predstavljajo posamezne pijače, stolpci pa parametre, ki jih posamezni pijači lahko nastavimo. Takšno rešitev uporablja točilni računalnik SKAT. Pri tem se določeni parametri medsebojno izključujejo.

Če želimo izvajati nadzorno in kontrolno funkcijo, mora točilni računalnik imeti realizirane prodajne ter bonirne števce. Najenostavnejša implementacija števcev je, da tabeli pijač za vsak števec dodamo po eno polje. Tako imamo vse podatke o pijači shranjene v isti tabeli in se na posamezen podatek sklicujemo preko ustreznega indeksa.

3.2.1.3 Vmesni pomnilnik

Nekateri sistemi imajo realiziran tudi vmesni pomnilnik za primere, če se prekine povezava s POS sistemom. Takrat govorimo o direktnem točenju. V vmesnem pomnilniku se shranjujejo sporočila točilnega računalnika, namenjena POS sistemu. Ko je povezava ponovno

vzpostavljena, POS sistem od točilnega računalnika zahteva vsebino vmesnega pomnilnika in mu tudi naroči, da ga sistem sprazni.

3.2.2 Delovanje točilnega računalnika

Vsaka pijača, ki jo na točilnem sistemu želimo točiti, ima pripadajočo tipko, ki je povezana s točilnim računalnikom. S pritiskom posamezne tipke tako računalniku pošljemo zahtevo za točenje. Ko računalnik zazna pritisk tipke, preveri v bonirnem števcu, če je pijača naročena. Če je, dopusti točenje, sicer ne. V primeru časovnega doziranja (opisano v razdelku 3.4 Načini doziranja pijač) nato preko polprevodniških stikal krmili ventil in iztoči nastavljeno količino. V primeru, da sistem za točenje uporablja turbino, je potrebna tudi komunikacija s turbino, ki pošlje impulz vsakič, ko naredi obrat. Ko število obratov doseže število nastavljenih obratov za pijačo, s čimer dosežemo željeno količino, računalnik zapre ventil in s tem konča točenje. Hkrati poveča prodajni števec pijače za 1.

Pomembno je omeniti, da kot pijačo ne razumemo le dejanske vrste pijače (npr. cola, sadni sok, voda itd.), ampak tudi ponavljajočo vrsto z nastavljeno drugačno količino. Namesto, da imamo za vsako vrsto pijače nastavljeno osnovno mero (npr. 1dcl) in nato za večje količine pritiskamo tipko večkratno, imamo lahko nastavljeno neko pijačo z najbolj pogostimi količinami in moramo tako tipko pritisniti le enkrat. V tem primeru imamo v tabeli pijač za vsako pijačo po eno vrsto, vse pa se točijo na istem ventilu. Iz tega izhaja prej omenjena zahteva po večjem številu vhodov kot izhodov v točilni računalnik. Torej, število različnih vrst pijač (npr. cola, sadni sok, voda itd.), ki jih sistem toči, nam določa število ventilov. Pri tem pa je pomembno omeniti, da število ventilov ni nujno enako številu pip, na katerih pijače točimo. Nekateri sistemi imajo za vsako pipo svoj ventil, drugi pa imajo po več ventilov povezanih na isto pipo. Zatorej le po številu pip točilnega sistema tudi ne gre soditi o številu pijač, ki jih sistem toči.

3.2.3 Lokacija točilnega računalnika

Iz prakse poznamo dve možni lokaciji točilnega računalnika:

- integriran v točilno napravo,
- samostojna naprava.

Delitev sem uvedel in omenjam zato, ker se točilni računalnik sistema SKAT, s katerim komunicira moja aplikacija, od ostalih sistemov najbolj razlikuje ravno v tem, da je »pakiran« v lastno ohišje in zaradi deluje kot samostojna enota. Ker so ostali proizvajalci večji in ponujajo širši spekter točilnih naprav, so primarno prilagojeni za povezovanje s točilnimi napravami lastne proizvodnje. Poleg tega pa njihovih točilnih računalnikov navadno ni mogoče uporabljati samostojno. Sistem SKAT se je za razliko od ostalih usmeril v povezljivost z raznovrstnimi napravami različnih proizvajalcev.

3.2.3.1 Integriran v točilni napravi

Pri integrirani različici, ki je bolj razširjena, imamo točilni računalnik vgrajen v glavo točilnega sistema. To ima svoje prednosti, saj imamo zaradi tega krajše povezave z

vhodno/izhodnimi napravami, ki so lažje dostopne. Prav tako se lahko kot konzola uporabijo kar zaslon in tipke točilnega sistema.

Rešitev na prvi pogled deluje dobro, vendar ima pomankljivost. Da jo bo mogoče bolje razumeti, si je potrebno predstavljati naslednjo situacijo. V nekem lokalu imamo zelo dolg točilni pult, ki ima (za naše namene število 2 že zadostuje) na vsakem koncu po en točilni računalnik. Oba točilna računalnika želimo povezati na POS sistem in preko tega kontrolirati iztočene pijače. Ker imamo opravka z dvema točilnima računalnikoma, imamo torej dve logični enoti. Ista pijača na obeh točilnih sistemih je med seboj ločena. Med njima ne obstaja povezava. To pomeni, da mora uporabnik, ki je boniral neko pijačo, razmišljati tudi, na katero logično enoto se bo naročilo zapisalo in bo pijačo tudi lahko iztočil le na tisti logični enoti. Takšna lastnost se zna v nekem nočnem lokalu z večjim prometom izkazati kot velika pomankljivost. Povezovanje takšnih sistemov v zaključene logične enote je sicer mogoče, vendar težje izvedljivo. Hkrati pa so tovrstni točilni računalniki prilagojeni za povezovanje s točilnimi napravami istega proizvajalca.

Iz zadnjega tudi sledi, da je sistem z vgrajenim točilnim računalnikom manj prilagodljiv oz. težje razširljiv. Če se odločimo za takšen sistem, ga bomo kdaj kasneje ob morebitnih razširitvah lokala težje dopolnjevali.

3.2.3.2 Samostojna naprava

Samostojna različica pomeni, da imamo točilni računalnik »pakiran« v lastno, od točilnega sistema ločeno, ohišje. Posamezne točilne naprave nato povežemo s točilnim računalnikom. Za vsako tipko vsakega posameznih sklopov točilnega sistema imamo na točilni računalnik narejeno eno povezavo. S tem dosežemo, da lahko na isti točilni računalnik povežemo več točilnih naprav, ki na enem računalniku delujejo kot zaključena logična enota. Prej opisana pomankljivost je s tem odpravljena. Prav tako je takšen sistem enostavneje modificirati, saj ga lahko v vsakem trenutku razširimo ali zožimo. Bistvena prednost tovrstnega točilnega računalnika pa je v tem, da lahko nanj povezujemo točilne naprave različnih proizvajalcev, saj točilni računalnik SKAT med njimi ne razlikuje.

3.2.4 Konfiguriranje točilnega računalnika

V razdelku 3.2.1 Opis točilnega računalnika je omenjeno, da točilni računalnik vodi tabelo pijač, kjer posamezna polja v tabeli predstavljajo parametre. Če bi točilni računalnik izdelali s prednastavljenimi lastnostmi, bi takšen sistem bil izrazito tog in po vsej verjetnosti neuporaben. Zaradi tega sistemi vseh proizvajalcev ponujajo tudi možnost konfiguracije sistema.

Pri tem je potrebno opozoriti, da navadno obstajajo različni nivoji dostopa. Uporabniški nivo je namenjen uporabniku in običajno omogoča le branje in brisanje števcev, v nekaterih primerih tudi spreminjanje kakšnega drugega parametra. Po drugi strani servisni nivo omogoča vse, kar omogoča uporabniški nivo ter tudi dostop do vseh ostalih nastavitev točilnega sistema.

Na sliki 3.3 je prikazan točilni računalnik SKAT. Slika je pomembna, ker je z nje razviden vmesnik točilnega računalnika, ki je glavni razlog za izdelavo aplikacije za konfiguriranje točilnega računalnika.

Slika 3.3: Točilni računalnik SKAT

Poznamo dva načina konfiguracije točilnega računalnika in s tem sistema:

- konzolno konfiguriranje,
- konfiguriranje preko PC.

3.2.4.1 Konzolno konfiguriranje

Praktično vsi proizvajalci ponudijo na svojem sistemu konfiguriranje s pomočjo konzole. Na glavni točilni napravi ali na točilnem računalniku imamo zaslon manjših dimenzij (npr. 5cm * 10 cm), ki zna prikazati majhno število vrstic s tekstno vsebino. Za premikanje po menijih uporabimo pripadajoče tipke (običajno le levo, desno, gor, dol, potrdi, prekliči). Tovrsten način je zelo tog, nepregleden ter zamuden, ker lahko zaradi omejene velikosti zaslona konfiguriramo le po eno pijačo naenkrat. Prav tako nam vnašanje vrednosti zaradi omejenega števila tipk vzame precej časa ter sistem lahko deluje nejasno, saj nimamo informacije o tem, na katerem nivoju v programu smo.

3.2.4.2 Konfiguriranje preko PC

V ta namen uporabimo povezavo točilnega računalnika na POS sistem. Če ima točilni računalnik realiziran konfiguracijski protokol, lahko izdelamo PC aplikacijo, s katero na preglednejši način predstavimo tabelo pijač ter omogočimo spreminjanje posameznih parametrov.

3.3 Funkcije točilnih sistemov

Glavne funkcije točilnih sistemov so:

- doziranje pijače,
- hlajenje pijač,
- nadzor in kontrola točenja pijač,
- prihranek skladiščnega prostora,
- priprava mešanih pijač.

3.3.1 Doziranje pijače

Točilni sistemi omogočajo natančnejše doziranje pijač, saj doziranje kontroliramo s pomočjo mehanskih ali elektronskih rešitev, ki so preciznejše od človeške roke. Predoziranje, ki je lahko posledica človeškega faktorja, namreč povzroči, da ob ugotavljanju stanja zalog ugotovimo manjko v zalogi. Ta nam povzroči dvojne stroške. Najprej na račun izgube materiala, ki smo ga plačali in ga nismo prodali. Nato še v obliki davščin, ki jih moramo poravnati pri prikazu inventurnega manjka. Precej problematičen vidik je tudi pojavitev dvoma v poštenje osebja. Obratno se nam brez dozirnega sistema tudi lahko zgodi, da ne iztočimo toliko, kolikor bi morali. Na hitro lahko ocenimo, da je to dobrodošlo, saj s tem, ko prodamo več, kot smo nabavili, povečamo svoj izkupiček. Dejansko pa je takšna ocena površna, saj ne vključuje nezadovoljstva strank, ki bodo prejkoslej ugotovile, da jih goljufamo, čeprav nenamerno. To bi seveda imelo negativen vpliv na dnevni promet.

3.3.2 Hlajenje pijač

Točilni sistemi ponujajo učinkovito hlajenje pijač. Brez točilnega sistema moramo pijače hladiti v hladilnikih. Pri tem običajno nimamo zadostnih hladilnih prostorov za pokrivanje celotnih zalog pijače, kar se lahko ob konicah izrazi v ponudbi neohlajenih pijač, s čimer zagotovo ne pridobimo novih strank. V primeru točilnih sistemov je lahko drugače.

Sokove imamo shranjenje v kontejnerjih v obliki sirupov, ki jih točilni sistem pomeša z vodo. Do mešanja pride v glavi točilne naprave. Voda se na poti skozi točilni sistem pretoči tudi skozi hladilni del sistema. Pri točenju je tako že ohlajena na željeno temperaturo.

Enako velja pri pivu in vinu, ki pa v točilni sistem vstopata že pripravljena. Oboje je običajno skladiščeno v neohlajenih sodih.

Na tem mestu velja še omeniti, da v točilni sistem vedno vstopa voda iz priključka za pitno vodo. Kljub temu sistem vodo tudi očisti s filtriranjem. Hkrati pa v sistemu lahko točimo tako negazirane kot gazirane sokove. Voda se v ta namen v primeru gaziranih sokov obogati še s CO2, kar se zgodi v »karbonatorju«. Produktu pravimo soda.

3.3.3 Nadzor in kontrola točenja pijač

Pri nadzoru točenja pijač odgovarjamo na naslednja vprašanja: Kdo sme točiti, kaj sme točiti in koliko sme točiti. Pod kontrolo točenja pijač pa se razume kontrola iztočenih količin in njihova primerjava z naročenimi količinami.

V primeru računalniško krmiljenih točilnih sistemov je točilni računalnik povezan s POS sistemom, ki delegira, kdo sme naročati ter od njega pridobi naročila za točenje. Možna je uvedba natakarskih ključavnic pri točilnem sistemu, s čimer se zagotovi tudi, kdo sme točiti.

Z uvedbo tovrstne kontrole in nadzora si lahko prihranimo marsikateri strošek, ki je običajno posledica ravnanja natakarjev, najsi gre za pijačo, ki jo natakarji sami spijejo, ali za pijačo, ki gre na »račun hiše«. Lahko pa gre tudi za najslabšo od možnosti, to je kraja s strani zaposlenih.

3.3.4 Prihranek skladiščnega prostora

V razdelku 3.3.2 Hlajenje pijač je bilo omenjeno, da v točilnih sistemih piva in vina skladiščimo v sodih ter sokove v obliki sirupov v kontejnerjih. Največji prihranek na prostoru je seveda pri sokovih, saj v obliki sirupa zasedejo bistveno manj prostora, kot če so polnjeni v steklenicah. Prihranek je vendarle tudi pri ostalih pijačah, saj manj prostora zasede sod piva kot enakovredna količina piva v steklenici. V tem primeru gre samo za prihranek na račun materiala, ki tvori steklenice, medtem ko pri sokovih gre tudi za prihranek na račun drugačnega agregatnega stanja, v katerem hranimo vsebino sokov.

3.3.5 Priprava mešanih pijač

Kadar imamo gostinski lokal, kjer se točijo večje količine mešanih pijač (npr. juice vodka), najpogosteje je to v nočnih gostinskih lokalih, nam točilni sistem lahko omogoča tudi pripravo mešanih pijač. Običajno imamo žgane pijače shranjene v steklenicah, ki so vezane direktno na dozerje. V tem primeru jih hranimo v kontejnerjih. Točilni sistem tako lahko na točilno pipo dostavi zmes večih pijač, ki so lahko kombinacija tako alkoholnih kot nealkoholnih pijač.

3.4 Načini doziranja pijač

Poznamo 3 osnovne načine doziranja:

- časovno doziranje,
- doziranje s pomočjo turbine,
- volumensko doziranje.

3.4.1 Časovno doziranje

Gre za način doziranja, pri katerem kontroliramo »odpiralni čas« ventila. To je tisti čas, ko je ventil odprt in toči pijačo.

Pri nastavljanju časa se izhaja iz enačbe (1) za prostorninski pretok.

*Prostorninski pretok = Površina * Hitrost [m3/s] (1)*

Površina je znana, saj so nam znane dimenzije ventila. Stalno hitrost dosežemo s konstantnim tlakom v ceveh. Ta se zagotavlja z uporabo reducirnih ventilov. Takšen način doziranja se uporablja pri točenju sokov.

Podoben sistem se lahko uporabi tudi pri točenju alkoholnih pijač iz steklenic. V tem primeru se na vrat steklenice namontira čep, s katerim kontroliramo »odpiralni čas«. Čep je enostaven magnet, ki ga odpiramo in zapiramo s pomočjo tuljave. Za razliko od zgornjega sistema v tem primeru nimamo zagotovljenega stalnega pritiska in s tem prostorninega pretoka. Pri točenju iz steklenice se to obrne na glavo in nam pritisk ustvarja količina pijače, ki je v steklenici. Ker se ta s časom spreminja, se hkrati spreminja tudi količina iztočene pijače. Sistem je tako le približek.

Obstajajo tudi modificirane verzije, ki uravnavajo tlak v steklenici, to pomeni, odpravljajo podtlak, ki se v steklenici pojavi od njenem praznjenju. Tako se zagotovi konstanten pretok in s tem natančnejša količina iztočene pijče.

3.4.2 Doziranje s pomočjo turbine

Drugi način je doziranje s pomočjo turbine. V tem primeru imamo pred ventilom namontirano turbino. Deluje tako, da ventil odpremo, »odpiralni čas« pa kontroliramo preko turbine. Za turbino imamo znan prostorninski pretok in s tem število obratov turbine, ki so potrebni za iztočenje željene količine pijače. Turbina število obratov točilnemu računalniku sporoča v obliki impulzov. Za vsak obrat pošlje impulz. Na točilnem računalniku imamo za pijačo nastavljeno število impulzov, ki določajo »odpiralni čas« turbine. Ko število impulzov doseže nastavljeno vrednost, se točenje konča.

Turbinsko doziranje se uporablja pri pivu in vinu. Da pivo lahko iztočimo iz soda, moramo tudi v tem primeru ustvariti pritisk, ki potisne tekočino. Pritisk ustvarimo s pomočjo CO2, katerega dovajamo v sod in z njim izpodrivamo pivo.

Stranski produkt tega je, da se pivo s časom nasiči s CO2, zaradi česar se hitreje peni. Hitrost raztapljanja CO2 v pivu je odvisna tudi od temperature. Zaradi penjenja piva pa želimo točilnemu osebju omogočiti spremembo pretoka, saj ga s tem lahko reguliramo. To deluje zato, ker »odpiralni čas« določa število obratov turbine in je ventil odprt do trenutka, ko se je iztočila natančna količina. Ne glede na to, koliko časa to je. V tem času lahko uporabnik hitrost točenja povečuje ali zmanjšuje in ga s tem optimizira.

Pri vinu težav s CO2 nimamo, saj ga z njim ne izpodrivamo, temveč ga črpamo s pomočjo membranskih črpalk.

3.4.3 Volumensko doziranje

Izraz volumensko doziranje sem uporabil, ker gre v tem primeru za doziranje s pomočjo posode, ki ima znan volumen. Gre za klasičen način doziranja pri žganih pijačah. V tem primeru steklenico z glavo navzdol posadimo v vhod dozerja na njegovem vrhu, na njegovem izhodu na spodnji strani pa točimo. Dozer deluje tako, se odprt vhod in odprt izhod medsebojno izključujeta. Ko s s spodnje strani pristavimo kozarec, smo s tem odprli izhod in

zaprli vhod. Pri tem se v kozarec iztoči količina iz posode. Ko kozarec odmaknemo, se izhod zapre in odpre vhod. Posoda se napolni s pijačo iz steklenice.

3.5 Ponudniki točilnih sistemov

Sedaj, ko so znane osnove točilnih sistemov, si na kratko oglejmo še nekaj glavnih ponudnikov tovrstne opreme v Sloveniji.

3.5.1 GRAPOS

Grapos je vodilni avstrijski ponudnik pijač, ki hkrati trži tudi točilne sisteme. Prodaja pijač in točilnih sistemov zanje gresta »z roko v roki« in mnogokrat so ponudniki pijač tudi proizvajalci točilnih sistemov.

Grapos sokove in točilne sisteme v Sloveniji ponuja podjetje Italcream, d. o. o. iz Maribora. Izraz »šankomat«, ki je postal žargonski izraz za imenovanje točilnega sistema, izhaja od Graposa, ki svojemu računalniško vodenemu točilnemu sistemu pravijo »Schankomat«. Na slovenskem tržišču so bili med pionirji.

Poznajo tako integrirano kot samostojno različico ter imajo implementiranih več rešitev, odvisno ali gre za večji ali manjši obseg točilnega sistema. Zlasti pomemben je njihov protokol za komunikacijo med POS sistemom in točilnim sistemom, ki se je z leti izkazal za izredno zanesljivega, hkrati pa je tudi enostaven. Za vsak ukaz, ki ga sistemu pošljemo, nam ta odgovori z »R« ali »F«, odvisno od tega, ali je ukaz bil sprejet ali ne. Ukazi po tem protokolu so dolgi med 4 in 8 byti. Zaradi enostavnosti in zanesljivosti ta protokol uporabljajo tudi nekateri slovenski proizvajalci. Povezava s POS sistemom deluje pri hitrosti 9600 baudov.

Verjetno imajo razvito tudi PC aplikacijo za konfiguriranje sistema, ki pa je na terenu ne uporablja niti njihov slovenski ponudnik, zato v praksi popravljanje konfiguriracije izvajamo preko konzole, ki je locirana na glavi točilnega sistema. To je pravzaprav edina slaba stran Grapos sistema.

3.5.2 NEKTAR

Blagovna znamka Nektar pripada podjetju Opoj sokovi, d. o. o. iz Komende. Podjetje ponuja tako sokove kot točilne sisteme in je kot takšno največji slovenski ponudnik na tem področju.

Gre za integrirano rešitev, za katero velja nekaj posebnosti. Doslej je veljalo, da vhode v točilni sistem predstavljajo tipke. V tem primeru namesto tipk nastopajo RFID skenerji. Rešitev je zaradi uporabe RFID tehnologije sicer naprednejša kot sama uporaba tipk, vendar ima tudi nekaj pomankljivosti. Z RFID skenerji se zagotovi drugi nivo kontrole točenja (enako kot zgoraj omenjene ključavnice). Če pri običajnih sistemih po tem, ko je neka pijača bila bonirana, le to lahko stoči vsakdo s pritiskom na ustrezno tipko, v tem primeru pijačo lahko stoči le oseba, ki poseduje RFID zapestnico. RFID zapestnice enolično identificirajo uporabnika. Če želi uporabnik iztočiti pijačo, mora ustrezni »tipki« približati zapestnico.

Slabost pri tej rešitvi je, da pritisk navadne tipke deluje vedno, če le ni pokvarjena. V primeru zapestnice pa se v določenih primerih lahko zgodi, da sistem ne prebere kode zapestnice uspešno že v prvem poizkusu, kar pomeni, da je lahko potrebnih tudi več poizkusov, s čimer se čas točenja po nepotrebnem podaljša. Hkrati pa je za doseganje zanesljivega branja zaradi lokacije zapestnice na roki lahko »mukotrpno«. V praksi gostinci to rešujejo tako, da imajo za točenje posebno zapestnico, ki je prosta in jo uporabljajo vsi natakarji, s čimer se dejansko izniči drugi nivo kontrole točenja. Ker zapestnice natakarji nimajo več locirane na roki, jo morajo običajno pred točenjem še poiskati, kar je realno korak nazaj.

Druga posebnost tega sistema je, da POS aplikacija ne komunicira direktno s točilnim sistemom, temveč komunikacija teče preko pomožne aplikacije, ki jo je potrebno namestiti na POS računalnik. Ta ustvari navidezni port, preko katerega POS aplikacija komunicira s točilnim sistemom. Vendarle pa ta aplikacija vsebuje tudi lični vmesnik za konfiguracijo sistema, s čimer se konfiguriranje sistema precej poenostavi.

Rešitev z zapestnicami je sicer ambiciozno zastavljena, vendar se je v praksi pokazalo nekaj pomankljivosti, predvsem počasnejše delo z uporabo zapestnic. Pravtako je takšna rešitev dražja, ker zahteva dodatno strojno opremo.

3.5.3 STARGAST

Stargast, d. o. o. se sicer ukvarja tudi s prodajo pijač, vendar je njihova ponudba usmerjena predvsem v tehnološke rešitve. V ponudbeni spekter spadajo oprema za toplotno obdelavo živil, za hlajenje živil, za pomivanje posode in tudi točilni sistemi. Skratka, gre za zaključeno ponudbo opreme, ki jo v gostinstvu potrebujejo.

Njihova rešitev točilnih sistemov je integrirana rešitev starejšega datuma, zaradi česar vsebuje veliko starejše strojne opreme, predvsem je potrebno omeniti počasnejše procesorje. Zaradi manjše procesorske moči so določene funkcije prenešene s točilnega sistema na POS sistem. V mislih imam predvsem bonirne in prodajne števce. Sistem v tem primeru deluje tako, da ko je na točilnem sistemu pritisnjena neka tipka, le ta POS aplikaciji sporoči, za pritisk katere tipke gre. POS aplikacija preveri vrednost bonirnega števca izbrane pijače in točilnemu sistemu s kratkim odgovorom sporoči, če ta pijačo sme stočiti ali ne. Če jo sme, POS aplikacija ustrezno poveča vrednost v prodajnem števcu.

Prednost njihove implementacije je v tem, da lahko z isto strojno opremo shajaš 15 ali več let, kar se ne odraža v menjavi generacij strojne opreme, v katere so prisiljeni ostali ponudniki. Slabost pa je, da če ne deluje povezava s POS sistemom, tudi ne delujejo kontrolne in nadzorne funkcije sistema. Točilni sistem je v v tem primeru potrebno postaviti v odprto stanje, s čimer je omogočeno prosto točenju vsakomur. Sistem opravlja le še funkcijo doziranja. Pri ostalih sistemih te funkcije nadzora in kontrole pri prekinitvi povezave s POS sistemom ostanejo vsaj v okrnjeni različici.

3.5.4 SKAT

Sistem Skat ni nova rešitev. Prve generacije so bile na trgu že pred leti, vendar je sistem zamenjal lastnika. Prvotni ponudnik se je iz posla umaknil in svojo rešitev prodal Bogme

Tehniki, k. d., ki jo sedaj nadalje razvija. Gre za najmanjšega izmed omenjenih ponudnikov. Sicer tržijo tudi lasten točilni sistem, vendar je poudarek na točilnem računalniku. Ta je za razliko od ostalih implementiran samostojno, s čimer omogoča povezovanje na raznorazne točilne sisteme. Je fleksibilen in poceni. Prednost ima v primeru majhnih rešitev, če želimo npr. imeti priklopljeno le 1x pivo in 1x kavomat. To pri integrirani implementaciji za realizacijo ni enostavno, saj ti proizvajalci navadno ne ponujajo tako omejenih točilnih naprav. Dodatna prednost pa izhaja iz fleksbilnosti, saj je z njim možno enostavneje spremememniti obstoječi sistem, najsi gre za nadgradnjo, razširitev ali redukcijo.

4 KONFIGURACIJSKI PROTOKOL DS1 TOČILNEGA RAČUNALNIKA SKAT

Izbira in določitev protokola je stvar izdelovalca točilnega sistema. V splošnem so si protokoli različnih proizvajalcev precej podobni, saj gre vedno za pošiljanje nekriptiranih (inkripcija ni potrebna!) tekstnih sporočil, katerih format je določen s strani izdelovalca. Sistemi nekaterih proizvajalcev poznajo tudi po več protokolov. Na tem mestu velja spomniti, da se je Grapos-ov protokol za naročanje in stornacije izkazal kot preprost in zanesljiv in ga tako poleg nekaterih drugih sistemov zna implementirati tudi sistem SKAT. Sistem SKAT ima sicer še svoj lasten protokol za naročanje in stornacije, ki se je s časom razvijal in je tako od BON1 verzije prešel že na BON3 verzijo.

Za namen konfiguriranja sistem SKAT uporablja konfiguracijski protokol DS1.

4.1 Protokol DS1

Komunikacija med točilnim računalnikom in POS sistemom oziroma poljubno PC enoto poteka po RS-232 kanalu. Na strani točilnega računalnika je v ta namen vgrajen moški DE9 konektor.

Parametri serijske povezave so 8-N-1. To pomeni, da je vsak znak, ki ga po kanalu pošiljamo, sestavljen iz 1 start bita, 8 podatkovnih bitov, 1 stop bita ter brez paritetnega bita.

Na sliki 4.1 je v predstavitev znaka prikazana v obliki diagrama.

Slika 4.1: Bitni okvir za pošiljanje enega ASCII znaka

Gre za asinhronsko komunikacijo, pri kateri se po kanalu pošiljajo znaki ASCII abecede. ASCII abeceda pozna 255 znakov, za kar potrebujemo 8-bitno besedo. Hitrost povezave je lahko 2400 ali 9600 baudov. Efektivna hitrost je manjša, saj se za podatkovne bite uporablja le 80 % vseh bitov, ki se po kanalu pošiljajo.

Točilni računalnik in aplikacija za konfiguriranje točilnega računalnika si po zgoraj opisanem kanalu izmenjujeta sporočila, katerih format je opisan v spodnjem razdelku.

4.1.1 Format sporočil

Sporočila, ki jih pozna točilni računalnik, so različnih dolžin ter imajo različen namen, zaradi česar točilni računalnik operira z več različnimi formati sporočil. Ločuje med formati sporočil za

- podajanje ukaza točilnemu računalniku za branje vrednosti,
- podajanje ukaza točilnemu računalniku za pisanje vrednosti,
- podajanje ukaza točilnemu računalniku za izvršitev operacije,
- odgovor točilnega računalnika v odvisnosti od (ne)uspešnosti izpolnitve ukaza.

Konec vsakega sporočila določa znak CR (Carriage Return), ki je v ASCII abesecedi predstavljen z znakom 13 (Dec).

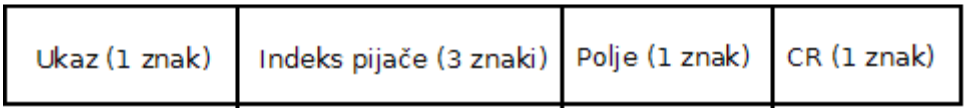
V spodaj predstavljenih formatih imamo za vrednost ukaza dve možnosti:

- R (Read; branje),
- W (Write; pisanje).

Z vrednostjo polja pa naslavljamo željeni parameter pijače.

4.1.1.1 Format ukaza za branje vrednosti

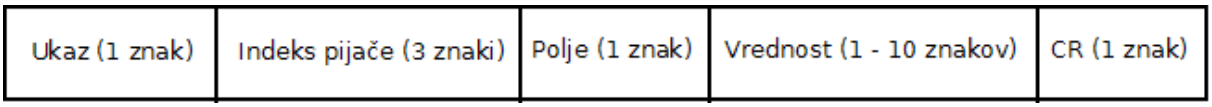
Pri branju moramo točilnemu računalniku sporočiti vrsto operacije (R), indeks pijače ter parameter, ki ga pri dani pijači želimo brati.



Slika 4.2: Format sporočila pri ukazu za branje vrednosti

4.1.1.2 Format ukaza za pisanje vrednosti

Format ukaza za pisanje poleg prej opisanih vrednosti vsebuje tudi vrednost parametra, ki ga želimo zapisati. Dolžina se razlikuje v odvisnosti od polja, ki ga zapisujemo. Dolžine posameznih polj so zapisane v 4.2 Parametri pijače.



Slika 4.3: Format spročila pri ukazu za pisanje vrednosti

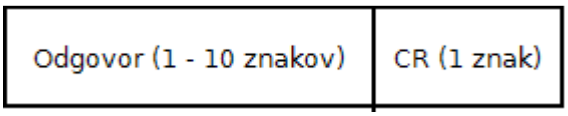
4.1.1.3 Format ukaza za izvršitev operacije

Format ukaza za izvršitev operacije je enak formatu ukaza za branje vrednosti, ki je razviden iz slike 4.2.

Točilni računalnik zna izvršiti dve operaciji, to sta brisanje vseh bonirnih števcev (vrednost Polje = H) ter brisanje vseh prodajnih števcev (vrednost Polje = G). Kot ukaz mu posredujemo vrednost W.

4.1.1.4 Format odgovora

Odgovor je lahko različnih dolžin, saj različna polja vsebuje vrednosti različnih dolžin. V primeru write ukaza je odgovor lahko ACK ali NACK. S prvim točilni računalnik sporoči, da je operacija bila uspešno opravljena, z drugim pa, da je prišlo do napake. V primeru branja odgovor vsebuje vrednost parametra, po katerem smo spraševali.



Slika 4.4: Format odgovora

4.2 Parametri pijače

V prejšnjem poglavju je bilo omenjeno, da delamo nad dvodimenzionalno tabelo, katere vrste predstavljajo pijače, polja v stolpcih pa posamezne parametre pijače. Vsaki pijači lahko nastavimo 11 parametrov. Točilni računalnik SKAT lahko naslavlja 130 pijač. Indeksi pijač se začno z 1 in končajo s 130. Parametre pijače z indeksom 0 pa sistem uporablja za nastavljanje parametrov povezave, zapis lastnika sistema ter zapis verzije programske opreme. Parametri, ki jih posamezni pijači lahko nastavljamo, so sledeči:

Naziv

Dolžina: 1 – 10 znakov; vrednost za olje: N

Parameter določa naziv pijače. Naziv se lahko izpiše na zaslonu točilnega sistema, sicer pa je pomemben le za nastavljalca sistema, da pri konzolnem konfiguriranju prepozna pijače.

Mode

Dolžina: 2 znaka; vrednost za polje: M

Parameter pove ali gre za turbinsko doziranje ali za navadno doziranje z ventilom.

Vhod

Dolžina: 2 znaka; vrednost za polje: I

S parametrom nastavimo indeks vhoda, na katerega je vezana ustrezna tipka.

Izhod1

Dolžina: 2 znaka; vrednost za polje: X

S parametrom nastavimo indeks izhoda, na katerega je vezan ventil, ki odpira pijačo.

Izhod2

Dolžina: 2 znaka; vrednost za polje: Y

Vsaka pijača je v tem sistemu lahko vezana na dva ventila. To pride prav pri mešanih pijačah, kjer lahko v primeru juice vodke nastavimo pijači, da odpira tako juice kot vodko.

Turbina

Dolžina: 2 znaka; vrednost za polje: D

Če je mode ustrezno nastavljen na turbinski režim, lahko s tem poljem določimo število impulzov turbine, ki določajo količino iztočene pijače.

Čas / število impulzov

Dolžina: 5 znakov; vrednost za polje: T

To lastnost uporabimo, kadar je mode časovno doziranje. Osnovna časovna enota je 100 ms, tako da čas nastavljamo dejansko nastavljamo kot faktor te enote.

Veza

Dolžina: 2 znaka; vrednost za polje: V

Omenjeno je že bilo, da v primeru dolgih točilnih pultov, ko je postavljenih več točilnih naprav, na katerih se toči ista pijača, želimo te naslavljati kot enotno logično enoto. To dosežemo z omenjeno lastnostjo, ki nam omogoča, da pijača uporablja bonirni števec druge pijače. Če bi npr. imeli dve točilni napravi za sokove in na vsakem želeli točiti jabolčni sok, bi jabolčnemu soku prve točilne naprave nastavili kot vezo indeks jabolčnega soka druge točilne naprave. Čeprav dejansko gre za isto pijačo, sta v logičnem smislu to dve pijači. Vsaka s svojim indeksom in vhodom ter izhodom.

Faktor

Dolžina: 2 znaka; vrednost za polje: F

Faktor se uporablja v povezavi z lastnostjo Veza. Najbolj razumljivo bo, če razložim na primeru. Imamo kavomat s 3 točilnimi glavami. Na vsaki lahko točimo enotni pijači, to je enojno in dvojno kavo. V tem primeru bi eno enojno kavo izbrali kot referenčno pijačo, ostalim enojnim kava bi nastavili kot vezo indeks te pijače. Pri dvojnih kavah pa bi poleg veze morali nastaviti tudi faktor, in sicer 2x, ker gre za dvokratno količino enojne kave.

Povezava / Pena

Dolžina: 3 znaki; vrednost za polje: P

Pri pivu se običajno uporablja tudi posebna tipka za peno. Sicer ne želimo spenjenega piva, vendarle pivo mora biti postreženo z nekaj pene. Ker vsaka tipka predstavlja vhod točilnega računalnika, v tem polju vpišemo indeks vhoda, na katerege je vezana tipka.

Izklop boniranja

Dolžina: 1 znak; Vrednost za Polje: Z

Kot izhaja iz imena, lahko s tem vklopimo/izklopimo možnost boniranja dane pijače, kar v praksi pomeni, da če boniranje izklopimo, smo pijačo s tem zablokirali, saj je preko POS sistema ni mogoče več bonirati.

4.3 Pomen polj pri pijači z indeksom 0

Pri pijači z indeksom 0 imajo pomen naslednja polja:

- Naziv pijače: lastnik točilnega sistema.
- Čas/število impulzov: število aktivnih artiklov.
- Veza: BAUDRATE (0 = 2400, 1 = 9600, 2 = 19200, 3 = 57600, 4 do 255 = 2400).
- Verzija (vrednost polje = v): verzija firmware.

Za verzijo točilnik računalnik uporablja dodatno polje (v) v dvodimenzionalni tabeli pijač. Ker pri konfiguraciji ni pomembna, jo omenjam le pri pijači z indeksom 0.

5 APLIKACIJA ZA KONFIGURIRANJE TOČILNEGA RAČUNALNIKA SKAT

Namen aplikacije je ponuditi prijaznejši, bolj logičen in hitrejši vmestnik za konfiguriranje točilnega računalnik SKAT. Kot je razvidno s slike 3.3, je vmesnik na točilnem računalniku zaslon z nekaj vrsticami in 6 tipkami, s katerimi kurzor premikamo po zaslonu. Konfiguracija več 10 artiklov, od katerih ima vsak 11 lastnosti, ki mu jih lahko nastavimo, je torej zamudna in nepregledna.

Aplikacija to delo olajša in hkrati omogoča, da si lahko pripravimo predhodno konfiguracijo, torej v fazi pred inštalacijo, kar skrajša čas inštalacije in zmanjša možnost nastavitve napačnih vrednosti, do česar lahko pride pri delu na terenu zaradi časovnih pritiskov, ki jim je inštalater izpostavljen.

5.1 Zahteve

Da bi aplikacija dosegla svoj namen, je bilo potrebno izpolniti naslednje zahteve:

- preverjanje dosegljivosti točilnega računalnika,
- samodejno iskanje COM porta, na katerem je priključen točilni računalnik,
- kreiranje prazne tabele za predhodni vnos konfiguracije,
- shranjevanje konfiguracije v XML datoteko,
- branje konfiguracije iz XML datoteke,
- branje, pisanje in brisanje enega ali več artiklov hkrati,
- branje, pisanje in brisanje bonirnega števca enega ali več artiklov hkrati,
- branje, pisanje in brisanje števca porabe enega ali več artiklov hkrati,
- branje in nastavljanje lastnika sistema,
- branje in nastavljanje števila aktivnih artiklov v točilnem sistemu.

Operacije branja in pisanja zahtevajo manipulacijo s serijskim portom. Pri teh operacijah gre za prejemanje in pošiljanje sporočil preko serijske povezave, ki morajo biti v skladu s protokolom, ki je opisan v pejšnjem poglavju.

Na istem principu temelji preverjanje dosegljivosti točilnega računalnika, pri čemer gre za pošiljanje zahteve za branje. Pri tem ni pomembno, katero informacijo beremo. Če odgovora ne prejmemo v določenem času, potem sklepamo, da točilni računalnik ni dosegljiv. Ali je računalnik izklopljen, nepovezan ali v kakšnem drugem stanju, s tem ne ugotovimo.

Pri samodejnem iskanju COM porta je ideja ta, da gremo v zanki skozi vse serijske porte na danem sistemu in pri vsakem preverimo dosegljivost točilnega računalnika. Ta operacija zahteva informacijo o serijskih portih, ki so nam na voljo. Pomembno je omeniti, da je operacija prilagojena OS Windows XP, saj podatek o tem, ali gre za dejanski ali za navidezni serijski port, išče v registru na prednastavljeni lokaciji, ki ni nujno enaka lokaciji, v kateri drugi verziji operacijskega sistema Windows. Razlogi so opisani v podpoglavju 5.3.4 Iskanje COM Porta.

Kreiranje prazne tabele je najenostavnejša operacija, saj je njena naloga le, da prikaže *DataGgridView* kontrolo z izbranim številom pravilno formatiranih praznih vrstic. Ko tabelo napolnemo s podatki, jo lahko ali zapišemo direktno na točilni računalnik, ali pa si lahko konfiguracijo shranimo v XML datoteko.

Pisanje in branje XML datoteke zahteva manipulacijo z XML. Potrebno je poznati le osnove jezika, saj je struktura zapisanih vrednosti preprosta. Gre pa za manipulacijo nad tekstno datoteko, v katero zapisujemo ali iz nje beremo vrstico po vrstici v XML formatu.

5.2 Izbrano orodje in okolje za razvoj aplikacije

Aplikacijo sem razvijal v programskem okolju Visual Studio 2008, izbral pa sem programski jezik Visual Basic.

Operacijski sistem, na katerem sem razvijal aplikaijo, je Microsoft Windows XP SP3.

Strojna oprema, na kateri sem razvijal aplikacijo, je prenosni računalnik HP Compaq nc8430 z Intelovim T7200 procesorjem (Centrino Duo, 2.0GHz) ter 2GB sistemskega pomnilnika. Na prenosnem računalniku so inštalirana ena serijska vrata ter več virtualnih serijskih vrat, ki so bile vir težave, na katero sem med razvijanjem aplikacije naletel. Detajlnejši opis težave je opisan v poglavju 5.3.4 Iskanje COM Porta.

5.3 Osnovne funkcije aplikacije

V razdelku 5.1 Zahteve so na kratko že bile omenjene ideje oziroma operacije, ki so potrebne za realizacijo zahtev. V tem razdelku bom izpostavil ključne procedure, ki so potrebne za dosego cilja, bodisi predstavljajo implementacijo celotne zahteve bodisi osnovne gradnike ostalih procedur.

Osnovne procedure/funkcije so naslednje:

- preverjanje dosegljivosti točilnega računalnika,
- iskanje COM porta,
- branje iz XML datoteke in pisanje vanjo,
- formatiranje ukaza,
- branje iz točilnega računalnika in pisanje vanj.

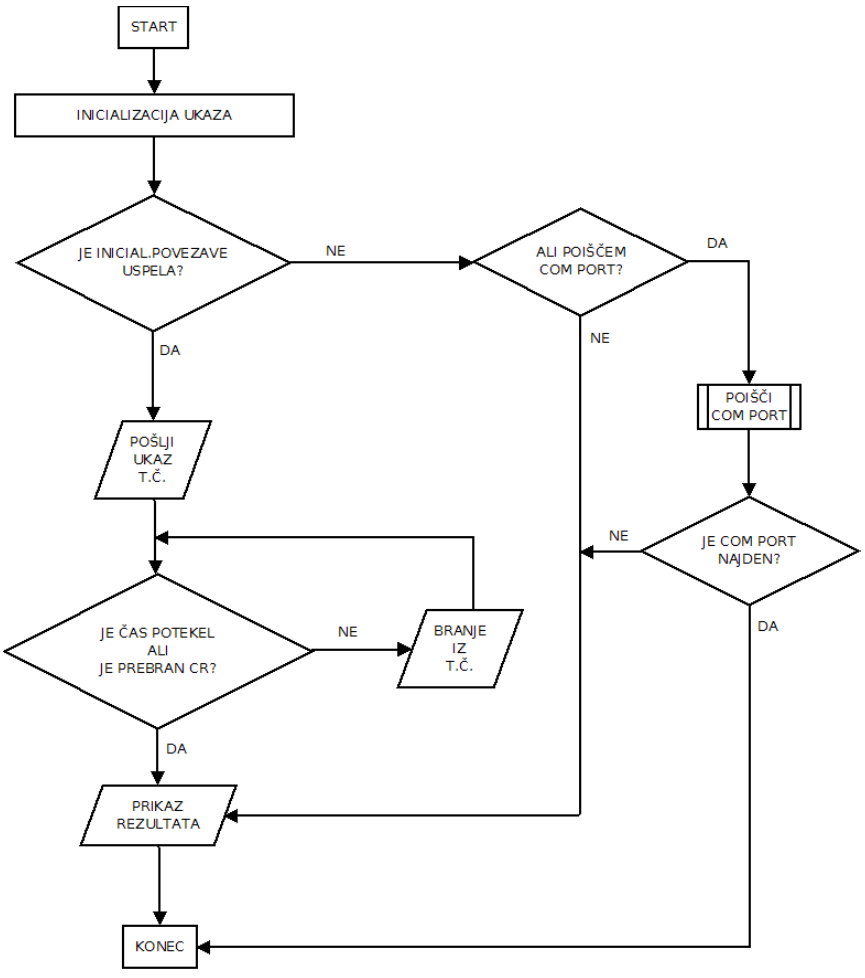
5.3.1 Preverjanje dosegljivosti točilnega računalnika

Dosegljivost preverjam z branjem posameznega podatka. Sam sem izbral število aktivnih artiklov.

Ukaz za preverjanje: R000T

Med razvojem aplikacije sem ugotovil, da bi bilo koristno, če bi pri določenih operacijah lahko preverjal dosegljivost točilnega računalnika, da lahko v skladu s tem omogočam in onemogočam posamezne opcije v menijih itd.. Praviloma bi to morala biti funkcija, ki vrača vrednost tipa boolean. Vendar, ker je ta informacija potrebna na več mestih in ker ne želim vsakič klicati te procedure, sem se odločil, da vrednost shranim v spremenljivko *BDATADosegljiva* tipa boolean, preko katere mi je informacija dosegljiva v vsakem koraku aplikacije.

Na sliki 5.2 je prikazan diagram procedure *PreveriDosegljivost*.



Slika 5.1: Diagram poteka procedure PreveriDosegljivost()

Deklaracija procedure je sledeča

```
»Public Sub PreveriDosegljivost(Optional ByVal ShowProgressBar As Boolean =
False)«
```

Procedura ima *Public* značaj, ker jo kličem tudi iz drugih form.Vhodni parameter sem določil, ker v nekaterih primerih želim prikazati napredovanje preverjanja dosegljivosti (npr. pri eksplicitnem klicanju te funkcije iz orodjarne).

V proceduri najprej preverim, če so serijska vrata odprta. Če niso, jih poizkusim odpreti . V primeru, da jih ne uspe odpreti, sklepam, da nastavljeni port ne obstaja in uporabnika vprašam, če želi, da aplikacija sama poišče port. Če pa vrata uspe odpreti, grem skozi isto proceduro kot pri *Read* in *Write*, to je formatiranje ukaza, pošiljanje ukaza, čakanje na odgovor.

Na koncu v skladu z vrednjostjo spremenljivke *rez* tipa boolean, katere vrednost določam v prejšnjih korakih, izpišem temu ustrezne informacije na ekran.

5.3.2 Iskanje COM porta

Avtomatizirano iskanje COM porta je dodatna funkcija, ki lahko olajša delo z aplikacijo. Kot je že bilo omenjeno, je osnovna ideja ta, da gremo v zanki skozi vsa serijska vrata na danem sistemu in pri vsakem kličemo proceduro *PreveriDosegljivost*, dokler ne najdemo pravih. Če aplikacija na nobenih vratih ne najde povezave do točilnega računalnika, je uporabnik o tem obveščen, prav tako tudi, če so vrata najdena.

Informacijo o serijskih vratih, ki so inštalirana na sistemu, dobimo preko `My.Computer.Ports.SerialPortNames`. Vendar nam ta preko tabele stringov posreduje le informacijo o številkah serijskih vrat, ki so nam na voljo (v obliki COMXX, kjer je XX številka vrat), ne pa tudi informacije o tipu vrat, to je, ali gre za navidezna (virtualna) ali dejanska vrata. Ko sem prvič poganjal zanko, mi je na prenosnem računalniku, kjer imam inštalirane tudi navidezne porte, pri pošijanju sporočil preko enih od teh vrat, generiralo neznano napako. Napake ni mogoče identificirati niti z `Try Catch` zanko. Aplikacija v tem primeru preprosto obvisi.

Rešitev sem iskal v registru sistema Windows, kjer bi morda našel dodatno informacijo o tipu serijskih vrat. Tega sicer nisem našel, sem pa na lokaciji »*HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\SERIALCOMM*« našel imena posameznih vrat v obliki *device\ime*. Zaradi tega je ta rešitev problematična, saj vse verzije sistema Windows registra nimajo povsem enako organiziranega. Ker sem aplikacijo razvijal v sistemu Windows XP, je možno, da na kakšni drugi različici aplikacija ne bo znala poiskati vseh informacij in posledično ne bi delovalo avtomatizirano iskanje COM porta. Tega nisem preizkušal. Vendarle to ni prevelika žrtev in se da brez te funkcije aplikacijo vendarle uporabljati z vsem, čemur je primarno namenjena.

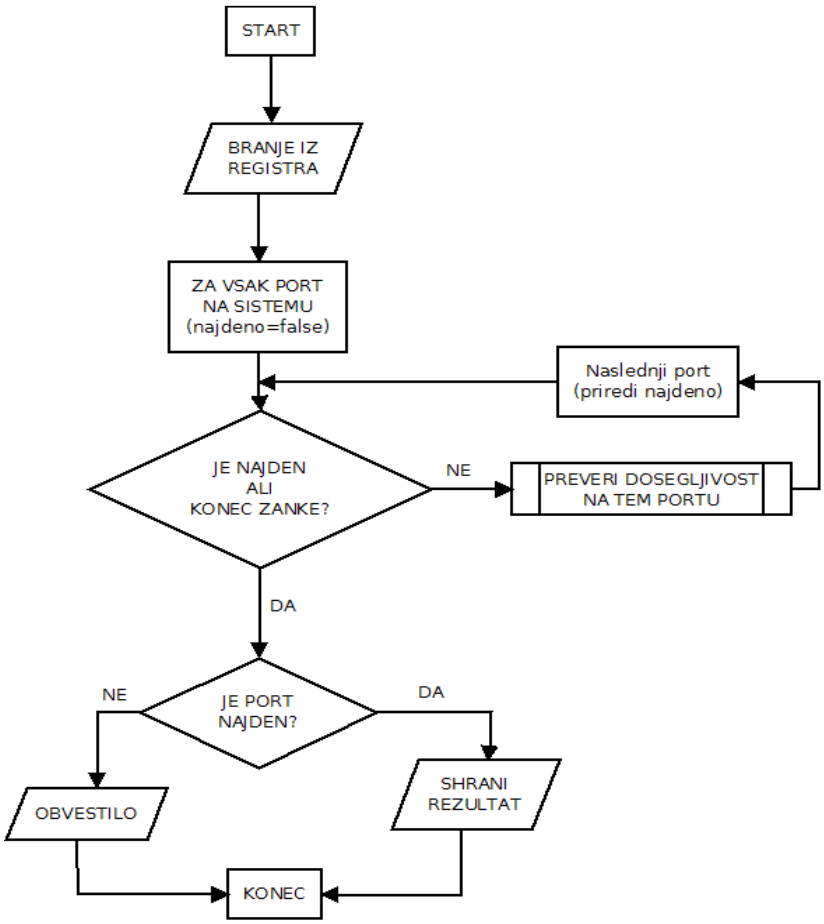
Deklaracije funkcije je sledeča

```
Public Function PoisciComPort(Optional ByVal SamoVrnemVrednost As Boolean =
False) As Integer
```

Kadar funkcijo kličem eksplicitno, to je s pritiskom ustreznega gumba na formi nastavitev (Public značaj), uporabnika obvestim o najdenih vratih in ga vprašam, če želi najdeno vrednost shraniti.

Drugi primer uporabe je, ko v *Load* proceduri preverjam, če ima aplikacija nastavljen port. Če ga nima, ga sama poišče in nastavi, brez potrjevanja s strani uporabnika. Za rešitev tega je namenjen opcijski paramater *SamoVrnemVrednost*.

Na sliki 5.2 je prikazan poenostavljen diagram *PoisciComPort* funkcije.



Slika 5.2: Diagram poteka funckije PoisciComPort()

5.3.3 Branje iz XML datoteke in pisanje vanjo

XML je razširljiv označevalni jezik, ki nam omogoča opisovanje podatkov v strukturirani obliki. Običajno se XML datoteke uporabljajo pri prenosu podatkov, sam pa sem jo uporabil pri shranjevanju konfiguracij. Vsako konfiguracijo, ki smo jo uredili v aplikaciji, je mogoče shraniti v datoteko in kasneje to datoteko odpirati in iz nje naložiti konfiguracijo v aplikacijo. Ta lastnost nam prinaša prednost v dveh primerih. V prvem primeru predstavlja prednost, ker je ob morebitnem sesutju točilnega računalnika možna hitrejša ponovna vzpostavitev

delujočega točilnega sistema. Zavedati se je potrebno, da se konfiguracija točilnega sistema lahko s časom spremeni. Že če ne drugega, smo doslej omenjali vsaj nastavitve točenja piva, ki jih lahko spreminja končni uporabnik. Zatorej obstaja verjetnost, da shranjena konfiguracija ne bo povsem ažurna, vendarle pa prihranimo nekaj časa, če nam je ni potrebno celotne ponovno spisati. Kot drugo je mogoče konfiguracijo pripraviti predhodno, torej pred samo inštalacijo, kar skrajša čas le-te.

Vsebina XML datoteke, ki jo aplikacija kreira, je sledeča

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Artikli>
	<Artikl>
		<StArtikla>X</StArtikla>
		<ImeArtikla>N</ImeArtikla>
		<Mode>X</Mode>
		<Vhod>X</Vhod>
		<Izhod1>X</Izhod1>
		<Izhod2>X</Izhod2>
		<Turbina>X</Turbina>
		<Cas>X</Cas>
		<Veza>X</Veza>
		<Faktor>X</Faktor>
		<Pena>X</Pena>
		<Izklop>X</Izklop>
		<Boniranje>X</Boniranje>
		<Stevec>X</Stevec>
	</Artikl>
</Artikli>
```

Glavno vozlišče imenujem *Artikli*, nasledniki tega vozlišča pa so posamezne pijače, ki jih nazivam *Artikl*. Vsak *Artikl* vsebuje vseh 11 omenjenih parametrov, prodajni in bonirni števec ter indeks pijače.

Tudi v tem primeru imam za vsako operacijo posebno proceduro. Za pisanje uporabljam spodnjo proceduro

```
Private Sub WriteToXml(ByVal FileName As String)
```

Proceduri posredujem samo ime datoteke, ki ga dobim s pomočjo *SaveFileDialog* kontrole.

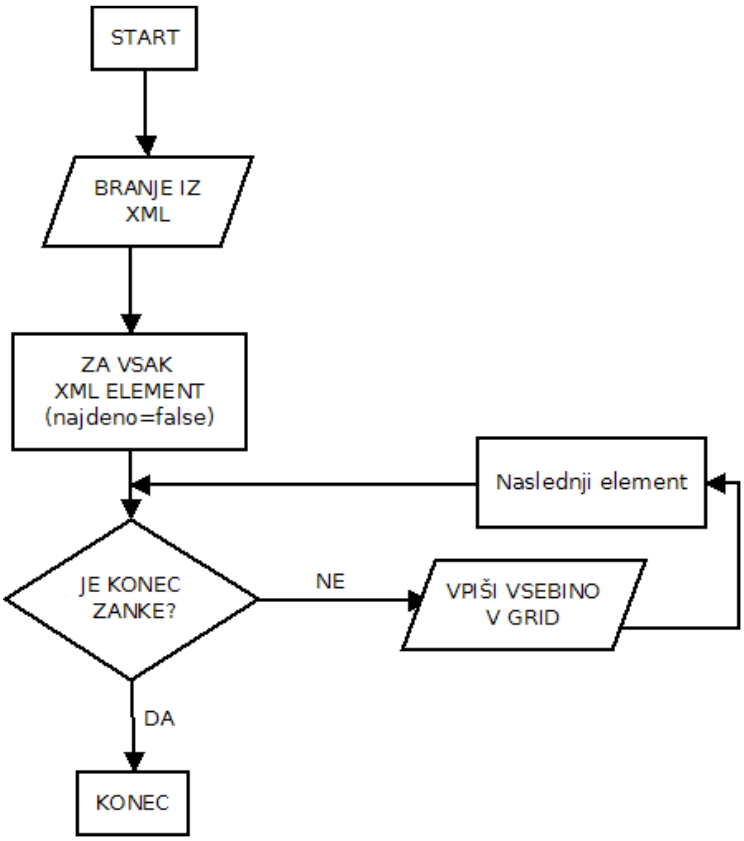
V proceduri najprej kreiram now *XmlDocument*, ki je podatkovna struktura za predstavitev XML dokumentov. Temu dokumentu določim deklaracijo, to je prva vrstica v zgornjem primeru. Nato grem s *for each* zanko skozi vse vrste *DataGridView* kontrole, v kateri imam zapisano konfiguracijo. Za vsako pijačo nato kreiram vozlišče Artikl z vsemi pripadajočimi parametri.

Za branje uporabljam proceduro

```
Private Sub ReadFromXml(ByVal FileName As String)
```

Tudi tej proceduri moram posredovati le ime datoteke, iz katere želim brati. Datoteko določim s pomočjo *OpenFileDialog* kontrole.

Na sliki 5.3 je prikazan diagram procedure *ReadXml*.



Slika 5.3: Diagram poteka procedure ReadXml()

Pri branju je postopek ravno obraten prejšnjemu. Začnem s kreiranjem *XmlDocument*, nakar grem v zanki skozi vsa vozlišča dokumenta in prebrano vpišem v *DataGridView* kontrolo. Pri tem moram za vsako prebrano pijačo kreirati novo vrsto v omenjeni kontroli.

Na tem mestu velja omeniti, da lahko vsebino *DataGridView* kontrole polnimo na dva načina. Eden je že bil omenjen, in sicer ročno dodajanje vrst, kjer moramo pri dodajanju posamezne vrste paziti, da stolpce polnimo s pravilnimi podatki. Drugi način bi bil, da kontroli kot podatkovni vir določimo (lastnost *DataSource*) tabelo podatkov. To bi storili s pomočjo objekta *DataSet*, ki omogoča predstavitev strukturiranih podatkov v sistemskem pomnilniku. V njem lahko torej hranimo podatkovne tabele.V tem primeru bi vse spremembe nad podatki delali v tabeli in nato osveževali *DataGridView* preko metod tega objekta. Ker obstajata tudi metodi *ReadXml* in *WriteXml*, bi si na ta način prihranil delo, ki ga opisujem v tem razdelku. Sam sem se odločil za prvi način, ker se izkaže za uporabnega, če npr. želim pri branju podatkov s točilnega računalnika sproti osveževati vsebino na ekranu ali pri kreiranju prazne konfiguracije.

5.3.4 Formatiranje ukaza

Namen procedure za formatiranje ukaza je, da pri branju in pisanju ni potrebno ponavljati kode za kreiranje ukaza. Hkrati je pomembna zato, ker implementira del zahtev, ki izhajajo iz protokola. Gre za funkcijo, ki ima 3 vhodne parametre in vrača vrednost, to je pravilno formatiran ukaz. Njena deklaracija je sledeča

```
Private Function FormatUkaz(ByVal Mode As RWModes, ByVal Polje As Char,
ByVal StArt As Integer, Optional ByVal Vrednost As String = Nothing) As
String
```

Razdelek 4.1.1 Formati sporočil govori o 3 različnih formatih sporočil. V aplikaciji imam v ta namen deklarirano enumeracijo *RWModes* z naslednjimi opcijami: pisanje z vrednostjo, pisanje brez vrednosti in branje. S parametrom *Polje* proceduri posredujem parameter, ki je vsebovan v ukazu. Parameter *StArt* določa pijačo. Oba sta obvezna. Zadnji parameter *Vrednost* je opcijski parameter, saj je uporabljen le pri načinu pisanje z vrednostjo.

Ker indeks števila zahteva tromestni zapis, *StArt* pa kot podatek tipa integer tega pogoja ne izpolnjuje, ga je potrebno konvertirati v parameter *strStArt* tipa string ter pri konverziji zagotoviti, da se v primeru manj kot 3- ali 2-mestne vrednosti prvotnega parametra zapišejo tudi vodeče ničle.

Pri pisanju z vrednostjo je na enak način potrebno obdelati tudi parameter *Vrednost*, odvisno od tega, ali gre za 2-, 3- ali 5-mestno vrednost. V primeru parametra naziv, kjer imamo opraviti z 10 znaki, je primeru manj mestne vrednosti potrebno dodati presledke, da dosežemo 10 mest.

Ukaz, ki ga funkcija tvori, je sledeče oblike

Ukaz = R/W + strStArt + Polje + Vrednost

5.3.5 Branje iz točilnega računalnika in pisanje vanj

5.3.5.1 Proceduri branja in pisanja

Branje in pisanje sta implementirana v procedurah *Read* in *Write*. Njuna koda je enaka do vključno branja odgovora točilnega računalnika, odtod naprej pa se razlikuje. Hkrati se razlikujeta v deklaraciji, saj mi pri bralni proceduri ni potrebno pošiljati parametra *Vrednost* ter *RWModes*, saj imamo za razliko od pisanja opraviti le z eno vrsto branja.

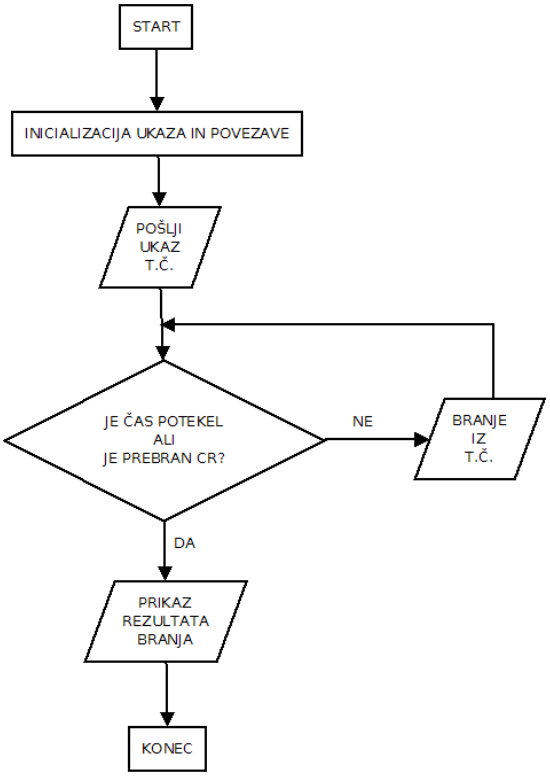
Njuna deklaracija je sledeča

```
Private Sub Write(ByVal Mode As RWModes, ByVal Polje As Char, ByVal StArt
As Integer, Optional ByVal Vrednost As String = Nothing)

Private Sub Read(ByVal Polje As Char, ByVal StArt As Integer)
```

Kot je razvidno iz deklaracije procedure *Write*, so parametri enaki kot pri funkciji za formatiranje. Ker ti dve proceduri predstavljata osnovne gradnike za večino operacij, s katerimi so realizirane zahteve, je bilo bistvenega pomena, da sem uspel spisati funkciji, ki jih lahko kličem pri vseh vrstah pisanja in branja na točilni računalnik in iz njega. Tako moram pri vseh tovrstnih operacijah razmišljati le še o tem, katero polje želim brati/nastavljati in za katero pijačo.

Na sliki 5.4 je prikazan poenostavljen diagram poteka *Read* procedure.



Slika 5.4: Diagram poteka procedure Read()

Osnovni princip obeh procedur je enak. Najprej formatiram ukaz na podlagi vhodnih parametrov. Ta ukaz nato pošljem preko serijskih vrat točilnemu računalniku, nakar zaženem »štoparico«. Ta je realizirana s pomočje *while zanke*. Zanka se zaključi, če je bil prebran znak <CR> ali če poteče maksimalni čas za branje, ki se ga nastavi v ini datoteki. Ker je za kompletno branje enega artikla potrebno izvesti 11 branj, je izbira tega časa pomembna. Premajhne vrednosti ne želimo nastaviti, prav tako tudi ne prevelike, ker bi v primeru več zaporednih neuspelih branj branje ene pijače trajalo predolgo. V primeru *Read* procedure po pretečenem času nadaljujem s prikazovanjem prebranega v odvisnosti od vrednosti. V primeru napake pri branju parametra (potekel čas ali odgovor *Nack*) pijače, ta polja v *DataGridView* kontroli označim z rdečo ali vpišem vrednost Nack. To sem naredil z namenom, da lahko razvijalec točilnega računalnika testira njegovo delovanje. Kar je bilo pravilno prebrano, zapišem na ustrezno mesto (DataGridView, StatusStrip). Pri pisanju pa v primeru neuspelega poizkusa zaključim proceduro in obvestim uporabnika o neuspelem pisanju.

5.3.5.2 Komunikacija pri različnih hitrostih

Točilni računalnik SKAT je trenutno sposoben komunicirati pri dveh hitrostih, to sta 2400 in 9600 baudov. Za v prihodnje predvideva tudi komunikacijo pri hitrostih 19200 in 57600 baudov, kar je razvidno iz vrednosti parametrov povezave, ki jih nastavimo preko pijače z indeksom 0.

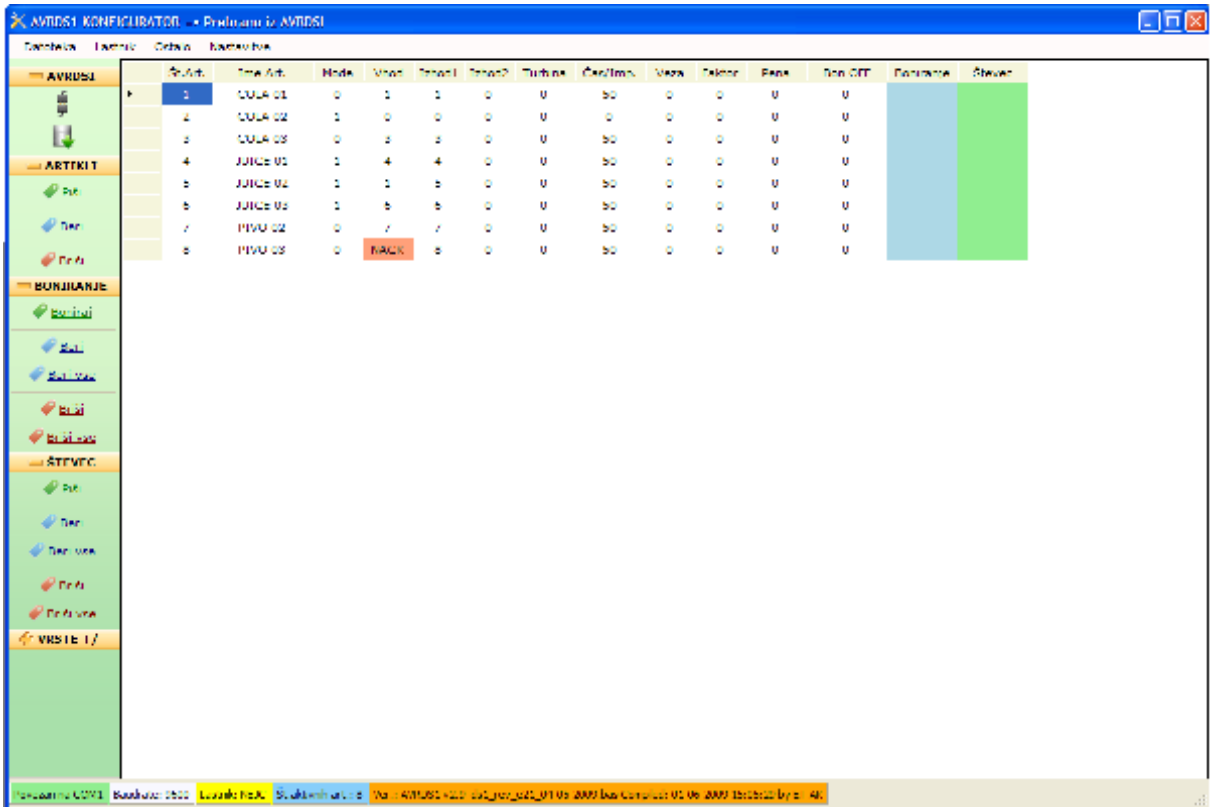
Med razvijanjem procedur za branje in pisanje sem testiranja opravljal pri obeh hitrostih. Testiranje prenosa pri hitrosti 2400 baudov je bilo brez napak. Ker sem z funkcijo za formatiranje ukaza poskrbel, da so poslani ukazi vedno pravilne oblike, in ker imam z grafičnim vmesnikom zagotovljene pravilne vrednosti (opisano v razdelku 5.4.3 Zagotavljanje pravilnega vnosa podatkov), načeloma odgovora NACK ne bi smel dobiti kot odgovor pri nobenem od možnih ukazov. Med testiranji se je izkazalo, da temu ni tako. Pri testiranju prenosa pri hitrosti 9600 baudov sta se pojavili dve napaki. Ali sem kot odgovor dobil NACK ali pa odgovora ni bilo in je *while* zanka postala neskončna, saj točilni računalnik ni poslal znaka CR, ki bi pomenil izhod iz zanke. Slednje je razlog, da sem kot pogoj za izhod iz while zanke uvedel tudi maksimalni čas branja. Rezultat reševanja težav pri komunikaciji z višjo hitrostjo je tudi uvedba, branju odgovora predhodnega praznjenja izhodnega vmesnega pomnilnika serijskih vrat, preko katerih aplikacija komunicira s točilnim računalnikom. Uvedel sem ga preventivno kot dodatno zagotovilo, da so ukazi, ki jih aplikacija pošilja, ter odgovori, ki jih aplikacija bere, pravilnih vrednosti. Po končanih testiranjih sem prišel do sklepa, da razlog za napake pri višji hitrosti nista proceduri, temveč mora njen izvor biti na točilnem računalniku.

Jedro točilnega računalnika SKAT je mikrokontroler AVR ATmega128 podjetja Atmel. Gre za 8-bitni mikrokontroler RISC arhitekture. Ima dvoje serijskih vrat (USART), ki pri frekvenci 11,0592MHz urinega signala mikrokontrolerja in 2400 baudih delujeta brez napak. Pri hitrosti 9600 baudov pa mikrokontroler pri istem taktu ni sposoben poganjati izhodnega vmesnega pomnilnika. Težava se je pojavljala, četudi mikrokontroler ni bil obremenjen in je ves procesorski čas posvetil delu s serijskimi vrati. Težava je bila odpravljena z dvigom frekvence urinega signala mikrokontrolerja na 14,047MHz, pri katerem serijska vrata delujejo brez napak vsaj do hitrosti 19200.

5.4 Grafični vmesnik

5.4.1 Opis

Grafični vmesnik aplikacije je prikazan na sliki spodaj. Sestavljajo ga 4 osnovni deli. Ti so menu na vrhu, orodjarna na levi, statusna vrstica na dnu in sredinski prikaz podatkov.



Slika 5.5: Grafični vmesnik aplikacije za konfiguriranje točilnega sistema SKAT

Preko menuja lahko dostopamo do opcij Nova konfiguracija, Odpri (konfiguracijo), Shrani (konfiguracijo), branje in pisanje lastnika, nastavljanje števila aktivnih artiklov ter do vmesnika z nastavitvami.

Prvotno sem preko menija ponudil tudi vse opcije, ki jih vidimo na levi strani v orodjarni. Ta način se je izkazal za neprikladnega, saj ni omogočal preglednosti nad opcijami in s tem hitrejšega dela. Zato sem na levo stran vključil orodjarno, ki vsebuje vse funkcije, ki so potrebne pri pripravi nove konfiguracije ter njenem branju in pisanju. Ikoni na vrhu orodjarne sta za preverjanje dosegljivosti točilnega računalnika in za branje celotne vsebine tabele pijač iz točilnega računalnika. K orodjarni se bom vrnil še v naslednjem razdelku.

Na spodnji stranici vmesnika je prikazana statusna vrstica, preko katere je uporabnik obveščen o stanju povezave, hitrosti povezave, lastniku, številu aktivnih artiklov ter verziji firmware-a na točilnem računalniku.

Na sredini je lociran glavni del aplikacije, to je vnosno polje. Za takšno prezentacijo podatkov sem se odločil iz dveh razlogov. Prvi razlog je, da je to najbolj naravni prikaz strukture tabele pijač na točilnem računalniku. Drugi razlog izhaja iz zahteve po hitrem delu. Vnašanje

podatkov je najhitrejše brez vmesnikov, direktno v *DataGridView* kontroli. O formatiranju vnešenih podatkov govori razdelek *3.5.3 Zagotavljanje pravilnega vnosa podatkov*.

Dizajniranje je pri vizualnem programiranju trivialno. Načeloma lahko vse postopke pri dizajniranju grafičnega vmesnika opravimo preko klikanja po lastnostnih posameznih objektov. To nam zadostuje, če je izgled vmesnika statičen. Kadar pa želimo izgled med izvajanjem aplikacije spreminjati, moramo spreminjanje zajeti v programu. V programerskem smislu sta pri dizajniranju aplikacije zanimiva dva primera. Prvi pri orodjarni, kjer želim vse gumbe barvno spreminjati z istim dogodkom. Drugi pri zagotavljanju pravilnega vnosa podatkov. V obeh primerih nalogo opravimo z ustvarjanjem lastnega upravitelja dogodkov (handler). Ta omogoča, da poljubni dogodek katerekoli kontrole povežemo z lastno proceduro, v kateri zajamemo odziv na klic dogodka.

5.4.2 Oblikovanje orodjarne

V orodjarni sem se, razen v primeru prvih dveh opcij, kjer bi besedilo bilo predolgo, odločil za kombinacijo besedila in ikon. Sestavljena je iz 5 sekcij. Vsako od teh sekcij je mogoče »odpirati« in »zapirati«. S tem sem imitiral prikaz drevesne strukture. To je bilo enostavno doseči. Potrebna je samo koda za prikazovanje in skrivanje posameznih sekcij z gumbi, ki se izvede ob kliku naslova posamezne sekcije, ki je predstavljen s *ToolStripLabel* kontrolo. Za vsako od teh sem obdelal dogodek *Click*.

V sklopih *Artikli*, *Boniranje* in *Števec* sem posamezne operacije glede na na njihovo namembnost ločil tudi barvno. Z namenom zagotavljanja večje preglednosti. Operacije za pisanje so zelene barve, operacije za branje modre barve ter operacije za brisanje rdeče barve. Ta nastavitev je statična, se torej ne spreminja med izvajanjem. Da bi še dodatno povečal preglednost aplikacije, sem se odločil, da bom v primeru, kadar kurzor lebdi nad katerim izmed gumbov, spremenil barvo ozadja tega gumba (lastnost *BackColor* od *ToolStripButton*). Spremenim jo v barvo besedila, vendar v svetlejši odtenek. Ko kurzor zapusti območje gumba, se barvno ozadje spremeni v prvotno. Da bi to dosegel, bi moral obdelati dogodka *MouseEnter* in *MouseLeave* za vsak posamezen gumb. Ker gre za 11 gumbov, bi bilo pisanje kode zamudno, hkrati bi se del kode ponavljal.

V takšnem primeru lahko ustvarimo lastnega upravitelja dogodkov. To pomeni, da spišemo proceduro, v kateri obdelamo določen dogodek in nato s klicem `AddHandler` nadomestimo prvotni dogodek. Sam sem napisal proceduri *TsBtnMouseEnter* in *TsBtnMouseLeave*, ki sem ju povezal z vsakim od naštetih gumbov. Ker se bo za vse gumbe izvajala ista procedura, je potrebno znotraj procedure nekako ločiti med gumbi, ki proceduro kličejo. Kot kriterij za razlikovanje med gumbi sem uporabil kar dele napisov, ki opisujejo, za katere vrste operacijo gre (branje, pisanje, brisanje). Pri novo napisani proceduri moramo le paziti, da je deklaracija procedure skladna z deklaracijo ciljnega dogodka. Če je deklaracija dogodka sledeča:

```
Private Sub TsBtnBonirajArtikl_MouseEnter(ByVal sender As Object, ByVal e
As System.EventArgs) Handles TsBtnBonirajArtikl.MouseEnter
```

je skladnost zagotovljena, če je deklaracija nove procedure naslednja:

```
Private Sub TsBtnMouseEnter(ByVal sender As Object, ByVal e As
System.EventArgs)
```

Uporabiti moramo torej iste argumente. Argument *sender* predstavlja objekt, ki kliče proceduro in preko njega spreminjam barvne lastnosti. Za konec tega razdelka prikazujem še stavek za dodajanje lastnega upravitelja dogodka neki kontroli.

```
RemoveHandler TsBtnBonirajArtikl.MouseEnter, AddressOf TsBtnMouseEnter
AddHandler TsBtnBonirajArtikl.MouseEnter, AddressOf TsBtnMouseEnter
```

Pravzaprav gre za dva stavka, saj je priporočljivo, da se morebitni obstoječi upravitelj dogodka najprej odstrani, šele nato doda nov. To izvedem v *Load* proceduri.

5.4.3 Zagotavljanje pravilnega vnosa podatkov

V tabeli pijač zahtevajo različna polja različne dolžine vrednosti. Pri nazivu pijače imamo 10 znakov, ki so lahko tako numerične kot nenumerične vrednosti. Pri ostalih poljih imamo od 1 do 5 numeričnih znakov. Odvisno od parametra.

Zagotavljanje dolžine je enostavno. *DataGridViewTextBoxColumn* ima lastnost *MaxInputLength*, s katero nastavimo maksimalno dolžino vnešenega teksta za posamezen stolpec. Ker moram po protokolu točilnemu računalniku vedno pošiljati toliko znakov, kolikor je predvidenih, se zgodi, da uporabnik vnese samo 1 znak v polje s predvidenimi 3 znaki. Kot je že bilo omenjeno, formatiranje tega podatka naredim v funkciji *FormatUkaz*, kjer zapišem vodeče ničle.

Ker imam v enem primeru dovoljene vse znake, v drugih pa samo numerične vrednosti, je potrebno v programu upoštevati tudi to. To je pomembno, da ne bi namesto številske vrednosti pošiljal točilnemu računalniku neštevilskih vrednosti. Da bi mi to uspelo realizirati, sem se usmeril v dogodke kontrole *DataGridView*. Izkaže se, da je pravi za to operacijo *EditingControShowing*. Ta se sproži vedno, kadar je prikazana kontrola za editiranje posamezne celice.

Deklaracija dogodka:

```
Private Sub dgv_EditingControlShowing(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewEditingControlShowingEventArgs) Handles
dgv.EditingControlShowing
```

Z uporabo *e.Control* se sklicujem na tekstno polje, ki se v času editiranja celice pojavi namesto celice. Če sedaj to kontrolo priredim kot vrednost objektu tipa *TextBox*, lahko z obdelavo *KeyPress* dogodka tega objekta nadzorujem pritisk vsake tipke nad tem objektom. Kreiranje novega objekta *TextBox* in prirejanje vrednosti izvedem vsakič, ko se sproži *EditingControlShowing*. Hkrati vsakič objektu dodam lasten upravitelj dogodka. Na enak način, kot je opisano v prejšnjem razdelku. Procedura, s katero obdelamo dogodek, je sedaj trivialna.

V proceduri se sprašujem po stolpičnem indeksu aktualne celice, saj po tem razlikujem med polji. V primeru *ImeArtikla* znak spremenim iz *lowercase* v *uppercase*. V primeru ostalih polj

pa se s funkcijo *IsNumeric* sprašujem, če je znak numerična vrednost. Edini preostali dovoljeni znak je *Backspace*. Če znak ne zadosti kriterijem, ne dovolim, da se izpiše.

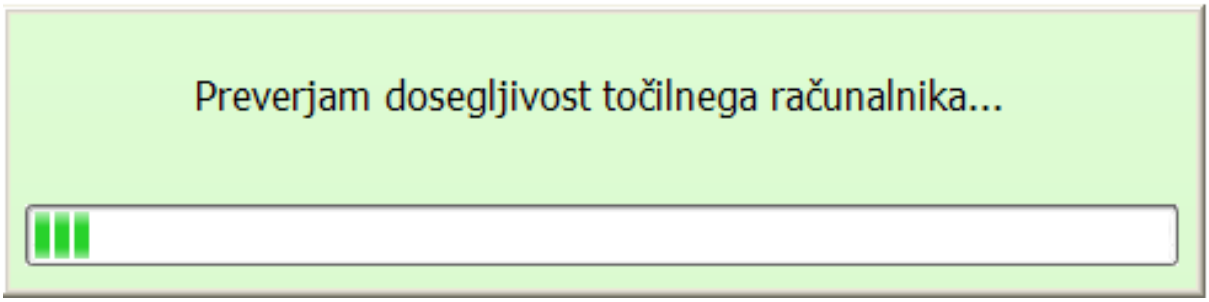
Spodaj je izpisana celotna procedura za obdelovanje dogodka:

```
Private Sub txtEdit_Keypress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs)
        'ImeArtikla ima lahko samo velike črke, ostali samo numeric
        If dgv.CurrentCell.ColumnIndex = 1 Then
            'ImeArtikla
            e.KeyChar = e.KeyChar.ToString.ToUpper
            e.Handled = False
        Else
            'Ostala polja
            If IsNumeric(e.KeyChar.ToString()) Or e.KeyChar =
            ChrW(Keys.Back) Then
                e.Handled = False
            Else
                e.Handled = True
            End If
        End If
End Sub
```

5.4.4 Forma SplashScreen

To je forma, ki se pojavi ob zagonu aplikacije, natančneje, prikazuje se med izvajanjem procedure *Load*. *SplashScreen* je uporaben, kadar ob zagonu aplikacije izvajamo časovno zahtevnejše operacije in se uporabniški vmesnik ne prikaže v trenutku. V takšnih primerih kličemo omenjeno formo, na kateri lahko npr. prikazujemo napredovanje zagona aplikacije.

V primeru moje aplikacije je forma bila potrebna, saj ob zagonu izvajam vrsto operacij, to so pripravljanje *DataGridView* kontrole, pripravljanje menuja in orodjarne, dodajam lastne upravitelje dogodkov ter časovno najzahtevnejša, preverjam dosegljivost točilnega računalnika. V primeru, da serijska vrata niso nastavljena, kličem tudi proceduro *PoisciComPort*.



Slika 5.6: Forma SplachScreen

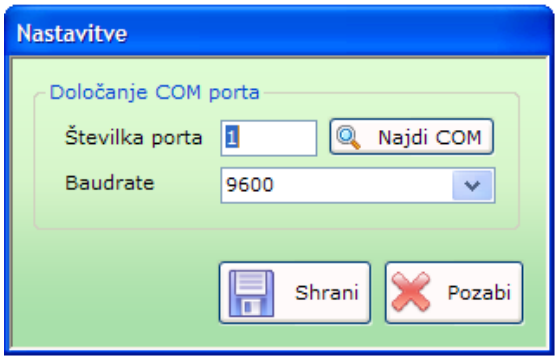
Podoba omenjene forme je prikazana na zgornji sliki 5.6. Ko je razvidno iz slike, na tej formi tudi prikazujem potek zagona aplikacije.

SplachScreen formo je priporočljivo uporabljati pri vsaki aplikaciji, saj se dolžina zagona aplikacij v .NET ogrodju razlikuje glede nato, ali gre za prvi zagon kakšne tovrstne aplikacije ali ne. Pri Microsoftu ločijo med t. i. hladnim in toplim zagonom. Pri hladnem zagonu ni bila

med delovanjem sistema pognana še nobena .NET aplikacija in traja zagon prve bistveno dlje časa kot v primeru toplega zagona. Ta čas gre predvsem na račun nalaganja množice podatkovnih knjižnic v delovni pomnilnik sistema, in sicer je »ozko grlo« tega procesa branje knjižnic z diska.

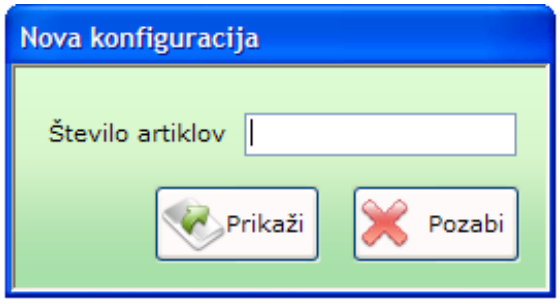
5.4.5 Ostale forme

Poleg že omenjenih dveh form uporabljam še 4 druge. Kot je razvidno iz slike 5.5, ima aplikacija v menuju tudi opcijo nastavitve. Nastavitve so skromne in vsebujejo le nastavljanje številke serijskih vrat ter baudrate hitrosti, na kateri bo komunikacija tekla. Sem pa sem vključil tudi že omenjeno možnost iskanja številke aktivnih vrat.



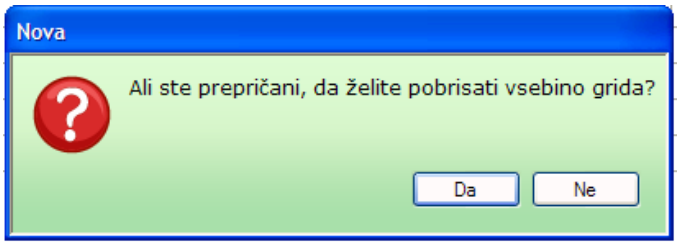
Slika 5.7: Forma z nastavitvami

V primerih, ko spreminjam lastnika ali število aktivnih artiklov, ter ko kreiram novo konfiguracijo, potrebujem formo, to je vnosno masko, preko katere bo uporabnik lahko vpisoval podatke. Ker gre v vseh primerih za vpisovanje enega podatka (lastnik, število aktivnih artiklov, število vrst v novi konfiguraciji), sem lahko uporabil isto formo za vse vnose. V odvisnosti od tega, za kakšen klic gre, formi ustrezno priredim napise in ikone. Na sliki 5.8 je primer vnosne forme pri kreiranju nove konfiguracije.

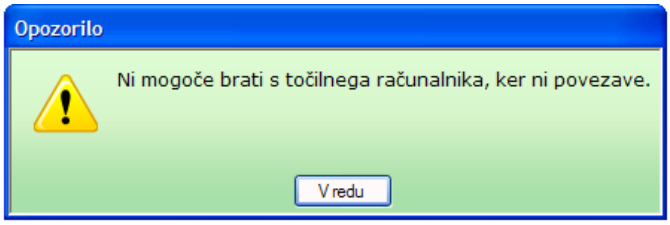


Slika 5.8: Forma za vnos enega podatka

Poleg omenjenih form vsaka aplikacija potrebuje še sporočilne forme, torej forme, preko katerih nekaj sporoča ali sprašuje. Ta forma se v Visual Basic-u imenuje MsgBox. Kot parametre mu lahko določimo besedilo, ki se bo izpisalo in vrsto sporočila. Kot je razvidno iz zgornjih dveh form, sta obarvani zeleno, ker sem želel doseči nekonvencioanlen izgled aplikacije. Zaradi tega sem ustvaril tudi lastno funkcijo *CustMsg*, katere klic je analogen klicu *MsgBox*. V funkciji se nato na podlagi vhodnega parametra, ki določa vrsto izpisa, odločim, ali prikažem *MsgBoxYesNo* ali *MsgBoxInfo*.



Slika 5.9: Sporočilna forma z vprašanjem



Slika 5.10: Sporočilna forma z opozorilom

5.5 Zagotavljanje odzivnosti vmesnika pri časovno zahtevnejših operacijah

Pri operacijah, ki trajajo dalj časa (npr. branje celotne tabele pijač iz točilnega računalnika), se zgodi, da se aplikacija neha odzivati na naše klikanje. Težava je v tem, da se aplikacija izvaja v eni niti, in ker je ta polno zasedena z branjem, ne more servisirati ostalih nalog. Lahko pa izkoristimo nitenje in branje izvajamo v ločeni niti. Potem se prvotna nit, v kateri teče uporabniški vmesnik, zopet odziva na naše ukaze. Brez tega so posledice očitne. Toliko časa kot traja v danem primeru branje celotne tabele (lahko tudi več minut), toliko časa se aplikacija ne odziva na druge ukaze. Če v tem času aplikacijo minimiziramo in nato ponovno fokusiramo okno, se nam vmesnik niti izriše ne več. Obenem je na takšen način nemogoče prekiniti operacijo, ker ni načina, da bi ji posredovali ukaz za prekinitev. Ta bi se izvedel šele po končani operaciji, ko je že prepozno. Prekinitev operacije pa je zaželjena opcija. V ta namen sem aplikacijo spremenil v takšno smer, da se vsi ukazi iz orodjarne izvajajo v ločeni niti.

5.5.1 Nitenje (threading)

Da bo mogoče bolje razumeti rešitev, ki sem jo uporabil v aplikaciji, je potrebno najprej spoznati osnove nitenja.

OS Windows (v nadaljevanju samo OS) omogoča vzporedno izvajanje več operacij/aplikacij hkrati. Tej lastnosti pravimo *Multitasking*. Da je to mogoče, mora OS imeti v sistemskem pomnilniku naloženo kodo in podatke več aplikacij hkrati, vendar jih mora med seboj ločiti. To stori z uvedbo *proces*-a. *Proces* je izolirano območje v pomnilniku, ki vsebuje kodo in podatke ene aplikacije. Če imamo opravka z enojedrnim procesorjem, potem je vzporedno izvajanje samo iluzija, saj OS procesorski čas dejansko razdeli na osnovne enote enakih dožin in nato vsakemu procesu omogoča izvajanje za čas osnovne enote. Ko prvi konča z delom, se

izvaja drugi itn., dokler ne pride na vrsto zopet prvi. Obstaja mnogo različnih strategij razvrščanja, na splošno strategiji v OS pravimo *TimeSlicing*. V primeru večjedrnih procesorjev se lahko v vsakem jedru izvaja po en proces. Razvrščanje tako postane še kompleksnejše.

Hkrati OS omogoča tudi *multithreading* oz. *nitenje*. To pomeni, da aplikacijo znotraj enega procesa razbijemo na več niti, od katerih vsaka izvaja svojo kodo. Zgoraj omenjena strategija pravzaprav razvršča niti in ne procesov (proces je le območje v pomnilniku, nit pa je dejanska izvajalna koda). Vse niti enega procesa si delijo pomnilnik, zaradi česar se zgodi, da lahko v nekem trenutku različne niti tekmujejo za iste resurse. Hkrati moramo paziti, da ne ustvarimo premalo ali preveč niti. Problem z enonitno aplikacijo je opisan na začetku razdelka. Preveč nit pa ni priporočljivih zato, ker je ustvarjanje niti procesorsko zahtevna operacija in bi s preveč nitmi procesor preobremenili. Zaradi tega je razvijanje aplikacij, ki tečejo večnitno, izrazito kompleksno. Moja aplikacija je pravzaprav enostavnejši primer. Uporabo niti npr. srečamo pri *Microsoft Word*, kjer v niti, ki se izvaja v ozadju, teče *SpellChecker* ali pa tiskanje. Nitenje je tudi pogost vzrok, da aplikacije obvisijo oziroma se nehajo odzivati. Najpogosteje zaradi tega, ker dve niti zahtevata hkraten dostop do iste pomnilniške lokacije. Če druga drugo izključujeta, se aplikacija neha odzivati. Takšni situaciji pravimo *deadlock*. Zato moramo pri razvoju večnitnih aplikacij nameniti izredno pozornost temu, da različne niti ne bi v istem trenutku dostopale do istih podatkov, in če do takšne situacije pride, moramo poskrbeti za strategijo, pri kateri se niti med seboj zmenijo, katera lahko dostopa do podatka.

Visual Basic omogoča več načinov za realizacijo nitenja v aplikaciji. Najpreprostejši način je uporaba *BackgroundWorker* class-a.

5.5.2 BackgroundWorker class

Backgroundworker pozna 4 dogodke. To so *DoWork*, *ProgressChanged*, *RunWorkerCompleted* in *Disposed*.

Kodo operacije, ki jo želimo izvajati v ločeni niti, napišemo v *DoWork*. *ProgressChanged* nas obvešča o napredku opreacije, *RunWorkerCompleted* pa se sproži, ko je *Backgroundworker* zaključil z delom.

Če imamo (izmišljeno) proceduro *PreberiPodatke*, potem bi običajno proceduro klicali ob sprožitvi dogodka neke kontrole, npr. *click* dogodek gumba. V tem primeru bi se koda izvedla v osnovni niti in bi vmesnik aplikacije ob njenem daljšem izvajanju postal neodziven. Če želimo to preprečiti, potem moramo klic procedure prestaviti v odziv na *DoWork* dogodek.

```
Private Sub BgdWorker_DoWork(ByVal sender As Object, ByVal e As
System.ComponentModel.DoWorkEventArgs) Handles BgdWorker.DoWork

      PreberiPodatke()

End Sub
```

Na mestu prvotnega klica procedure bi sedaj klicali *Backgroundworker* metodo R*unWorkerAsync*. S tem sprožimo *DoWork*. Prekinitev izvajanja dosežemo s klicem metode *CancelAsync*.

V mojem primeru je koda daljša, ker uporabljam en Backgorundworker in sem v *DoWork* moral vključiti odločitveni stavek `Select Case`, s katerim se glede na prejeti argument (argument posredujemo pri klicu *RunWorkerAsync*) odločam, za klic katere procedure gre. Med izvajanjem procedur, ki se izvajajo na takšen način, v statusni vrstici ponudim prikaz napredka izvajanja ter gumb *Prekini*. Branje ene vrste je sicer dovolj hitro, da nam ni potrebno poseči po tem gumbu, pri branju več vrst ali celotne tabele pa pride prav. V tem primeru branja ne prekinem med branjem vrste, temveč šele po končanem branju celotne vrste.

Tako izgleda prikaz napredka in gumb *Prekini* v statusni vrstici.

Slika 5.11: Prikaz statusa napredka ter gumb Prekini

6 SKLEPNE UGOTOVITVE

Teoretični del diplomske naloge je predstavljen z vidika razvijalca POS sistemov. Poleg opisa splošnih lastnosti točilnih sistemov je poudarek predvsem na razumevanju delovanja točilnega sistema, kot ga mora razumeti ravijalec POS sistemov, če želi zagotavljati povezljivost POS sistema z računalniško krmiljenim točilnim sistemom. Literatura s tega področja je skopa, zaradi česar se po meni dostopnih informacijah večina napisanega prvič pojavi v slovenski literaturi.

Med pisanjem aplikacije sem naletel na eno večjo težavo, in sicer napako pri komunikaciji pri hitrosti 9600 baudov. Kot se je izkazalo, je vzrok za težavo izhajal iz delovanja točilnega računalnika in je tudi že odpravljen.

Pomemben vidik zame je, da sem se pri izdelavi aplikacije moral poglobiti v tematiko, zaradi česar sem pridobil precej novega znanja z danega področja, ki mi bo vsekakor koristilo pri prihodnjem delu. To je tako s področja delovanja točilnih sistemov kot tudi s področja povezljivosti točilnih sistemov z blagajniškim sistemom ali katerim drugim računalniškim sistemom. Na razvoj aplikacije so vplivale tudi v praksi pridobljene izkušnje, saj so končni obliki botrovale pripombe uporabnikov aplikacije na terenu.

Aplikacijo bi se dalo nadgraditi v smeri, da bi omogočala konfiguriranje na daljavo. Dandanes, ko je internetna povezljivost na visokem nivoju, je izvedba tega precej preprosta. Ker točilni računalnik ne omogoča povezovanja v internet, bi se v ta namen v internet povezalo blagajniški računalnik, ki ima povezavo s točilnim računalnikom. Blagajniški sistemi našega podjetja imajo tovrstno povezljivost zagotovljeno, kjerkoli obstaja internetni priključek, saj nam oddaljeno povezovanje omogoča, da se marsikatera težava odpravi na daljavo. Potrebno bi bilo torej spisati servisno aplikacijo, ki bi na blagajniškem računalniku deloval kot server, na katerega bi se povezovala aplikacija za konfiguriranje. Servisna aplikacija pa bi s točilnim računalnikom komunicirala preko obstoječe fizične povezave. Zaenkrat ostajam samo pri ideji, morda pa se bo kdaj v prihodnosti pokazal interes proizvajalca točilnega sistema, da bi se aplikacija razširila v tej smeri.

7 PRILOGE

7.1 Dodatek A: Slike točilnih naprav

Slika 7.1: Točilne pipe za pivo točilnega sistema Schankomat

Slika 7.2: Točilne pipe za sokove in vino točilnega sistema Schankomat

Slika 7.3: Schankomat TABC sistem za točenje žganih pijač

Slika 7.4: Omare za skladiščenje steklenic sistema Schankomat

Slika 7.5:Kavni avtomat podjetja La Scala

7.2 Dodatek B: Kazalo slik

8 LITERATURA

[1] Evjen B., Hollis B., Sheldon B., Sharkey K., *Professional Visual Basic 2008*, Wiley Publishing, 2008

[2] Petroutsos E., *Mastering Microsoft Visual Basic 2005*, Wiley Publishing, 2006

[3] Jones P., *Visual Basic .NET*, Continuum, 2003

[4] Mesojedec U., VISUAL *BASIC sodobno programiranje*, Pasadena, 1998

[5] DePetrillo B. A., *Razumeti Mircosoft .NET*, Pasadena, 2002

[6] (2009) Spletna stran Microsoft. Dostopno na:

http://www.microsoft.com

[7] (2009) Spletišče Codeguru. Dostopno na:

http://www.codeguru.com

[8] (2009) Spletišče The Code Project. Dostopno na:

http://www.codeproject.com

[9] (2009) Spletišče Experts Exchange. Dostopno na:

http://www.experts-exchange.com/

[10] (2009) Spletna enciklopedija Wikipedia: Point of sale. Dostopno na:

http://en.wikipedia.org/wiki/Point_of_sale